View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Information and Computation 188 (2004) 143–178

Information and Computation

www.elsevier.com/locate/ic

# Decidability of bounded second order unification

Manfred Schmidt-Schauß*

*Institut für Informatik, Johann Wolfgang Goethe-Universität, Postfach 111932, Frankfurt D-60054, Germany*

Received 11 November 1999

**Abstract**

It is well-known that first order unification is decidable, whereas second order (and higher order) unification is undecidable. Bounded second order unification is second order unification under the restriction that only a bounded number of bound variables in the instantiating terms for second order variables is permitted, however, the size of the instantiation is not restricted. In this paper, a decision algorithm for bounded second order unification is described. This is the first nontrivial decidability result for second order unification, where the signature is not restricted and there are no restrictions on the occurrences of variables. This supports the claim that bounded second order unification is easier than context unification, whose decidability is currently an open question.
© 2003 Elsevier Inc. All rights reserved.

*Keywords:* Unification; Second order unification; Lambda calculus; Context unification; Automated deduction; Rewriting; Higher order unification; Logics in artificial intelligence

## 1. Introduction

Unification is solving equations. In automated deduction systems based on first order predicate logic, in particular in logic programming, (first order) unification is a central operation. It is the question whether two terms formed from function symbols and variables can be made equal by instantiating the variables by terms. It is well-known that first order unification is decidable in linear time (see the overview article [1]).

Second order unification (SOU) generalizes first order unification by extending the notion of terms with second order variables that may be instantiated with lambda-terms of the form $\lambda x_1, \ldots,$

*Corresponding author. Fax: +49-69-798-28919.
*E-mail address:* schauss@ki.informatik.uni-frankfurt.de.

0890-5401/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2003.08.002

$x_n.t$, where $n$ is the arity of the second order variable, and $t$ is a lambda-free term. Second order unification is a specialization of higher order unification [12,25,26].

Goldfarb [10] has shown that second order unification is undecidable. This result was sharpened in [7] for a restricted signature and in [13,14] for severe restrictions on occurrences and arities of second order variables.

Monadic second order unification (MSOU) (see [8]) is second order unification where the signature has only constants and monadic function symbols, i.e., there are no function symbols of arity 2 or more. Its decidability follows from decidability of string unification [16]. Recently obtained upper bounds for the complexity of string unification are EXPSPACE [11], which was improved to NEXPTIME [17] and even to PSPACE [18].

Another kind of restriction which has recently attracted some interest is to permit as instantiations for the second order variables only terms where the number of occurrences of bound variables is in a prescribed set. If the number is exactly one and the second order variables are unary (i.e., of arity 1), then this is called context unification. It is currently an open question, whether context unification is decidable.

If context unification is restricted, then there are some results on decidability: if the number of context variables is at most two [19–21]; if the nesting of second order variables has a certain form [22–24]; if for every context variable $X$, all occurrences of $X$ have the same argument [3,4]; or if every variable and every context variable occurs at most twice [15].

In this paper we consider the problem of *bounded second order unification* (BSOU) where the total number of occurrences of bound variables in the instantiation of second order variables is bounded, but zero is also permitted. A specialization is Z-context unification, where the second order variables are unary, and the number of occurrences of bound variables in an instantiation of a second order variable (Z-context variable) may be zero or one, whereas context unification permits only instantiations of unary second order variables (context variables) with a term having exactly one occurrence of a bound variable.

The main part of this paper is to construct a decision algorithm for Z-context unification. It is shown that decidability of BSOU is equivalent to decidability of Z-context unification problems. This implies the new result that bounded second order unification is decidable; as a lower complexity bound, it is shown that BSOU and MSOU are $\mathcal{NP}$-hard. MSOU is already known to be decidable [8] by using a decision algorithm for string unification. This paper provides an alternative decision method for MSOU.

The result on decidability shows that a slight restriction makes second order unification decidable, which has a potential usage in implementations. A semi-decision procedure can easily be built upon such a decision algorithm by successively increasing the bound on the number of occurrences of bound variables.

The relation to the open problem of decidability of context unification is as follows. Decidability of context unification would imply decidability of the bounded second order unification problem. Nevertheless, the algorithm given in this paper would remain of interest, since it is independent of [16,18], whereas a (hypothetical) decision algorithm for context unification must also solve the string unification problem.

The decision algorithm for BSOU cannot be turned into a decision algorithm for context unification, since a final (pre-unified) unification problem wrt. bounded unification may be not unifiable

as a context unification problem. It is completely unclear how to decide whether a final problem is unifiable w.r.t. context unification.

Thus, second order unification with a restriction on the number of occurrences of bound variables splits into two apparently different problems: one is bounded second order unification, the other is context unification. The basic difference is whether zero occurrences of bound variables are permitted in the instantiation or not.

Bounded second order unification can be seen as an instance of a general method to identify a parameter of a problem that helps estimating the complexity of a problem slicewise, depending on the parameter, as treated in [6].

The paper is structured as follows: after explaining the notation in Section 2, in Section 3 bounded second order unification is shown to be equivalent to Z-context unification. A decision algorithm for Z-context unification is constructed in Sections 4, 5, 6. Section 7 contains a proof that bounded and unary second order unification are $\mathcal{NP}$-hard. The appendix contains proofs of the key lemmas of Section 6.

## 2. Preliminaries

Let $\Sigma$ be a signature of function symbols. Every function symbol comes with an arity, denoted $ar(f)$, which is a nonnegative integer. Function symbols with $ar(f) = 0$ are also called *constant symbols*. We assume that the signature contains at least one constant symbol, in particular we also allow that the signature may be infinite or monadic.[1] Let $\mathcal{V}_1$ be the (infinite) set of first order variables, let $\mathcal{V}_{2,i}$ be the (infinite) set of second order variables of arity $i$, and let $\mathcal{V}_2 := \bigcup \mathcal{V}_{2,i}$. First order variables are denoted by letters $x$, $y$, and $z$, second order variables by letters $X$, $Y$, and $Z$, and if we mean first or second order, then we use $\mathcal{X}$, $\mathcal{Y}$, and $Z$. The arity of a second order variable $X$ is denoted as $ar(X)$.

*Terms* $t$ are formed using the grammar

$$t ::= x | f(t_1, \ldots, t_{ar(f)}) | X(t_1, \ldots, t_{ar(X)}),$$

where $x$ is a first order variable, $f$ is a function symbol, $X$ is a second order variable, and $t_i$ are terms. For a constant symbol $a$, we write $a$ instead of $a()$. We denote terms using the letters $s, t$. Syntactic equality of terms $s, t$ is denoted as $s \equiv t$. If a term does not contain second order variables, then we say call it a *first order term*. The set of variables occurring in the term $s$ is denoted as $Var(s)$. We call $f$ the *head* of the term $f(t_1, \ldots, t_{ar(f)})$, $X$ the head of the term $X(s)$, and $x$ is the head of $x$. We will use tree addresses for pointing to subterms in a term, and call them positions. The subterm of $t$ at position $p$ is denoted as $t_{|p}$. The notation $t[s]_p$ means that $t$ has a subterm $s$ at position $p$. A term $s$ is called a *ground term* if $s$ has no occurrences of variables. The number of occurrences of a variable $x$ in a term $t$ is denoted $occ(x, t)$.

To define substitutions we will use an (untyped) lambda-notation, where the bound variables are always first order variables, and the body is a ($\lambda$-free) term. A *ground substitution* is a mapping from terms to ground terms, which is determined by its values on variables. Since we only

[1] Monadic means that every function symbol has arity 0 or 1.

use ground substitutions, we will only speak of substitutions in the following. A substitution $\sigma$ can be represented as $\{x_i \to t_i \mid i = 1,\ldots,n\} \cup \{X_j \to L_j \mid j = 1,\ldots,m\}$, where $t_i$ for $i = 1,\ldots,n$ is a ground term and $L_j$ for $j = 1,\ldots,m$ is a closed lambda expression $\lambda x_1,\ldots,x_{ar(X_j)}.s_j$, where $s_j$ is a first order term, and only the variables $x_i$ for $i = 1,\ldots,ar(X_j)$ can be free variables in $s_j$. The *domain* of $\sigma$ is the set $\{x_i \mid i = 1,\ldots,n\} \cup \{X_j \mid j = 1,\ldots,m\}$, and the *codomain* of $\sigma$ is the set $\{t_i \mid i = 1,\ldots,n\} \cup \{L_j \mid j = 1,\ldots,m\}$. The substitution $\sigma$ operates on a term $t$ by replacing all occurrences of variables $x_i$ by $t_i, i = 1,\ldots,n$, and replacing all occurrences of second order variables $X_j$ by $L_j, j = 1,\ldots,m$, followed by $\beta$-reductions. We tacitly assume that a substitution $\sigma$ is only applied to terms $t$ where $Var(t)$ is a subset of the domain of $\sigma$, hence we assume that the result of applying a ground substitution to a term results in a ground term.

We use the notations $Id$ for the expression $\lambda x.x$, and $(K\ s)$[2] for the expression $\lambda x.s$, where $s$ is a first order term that does not contain $x$. We usually assume that an instantiation $Id$ or $(K\ s)$ for a second order variable in a term is immediately simplified: $Id(t)$ is replaced by $t$, and $(K\ s)(t)$ by $s$.

*Contexts* are formed by the grammar[3]

$$C[\cdot] ::= [\cdot] \mid X(C[\cdot]) \mid f(t_1,\ldots,t_{j-1},C[\cdot],t_{j+1},\ldots,t_{ar(f)}),$$

where $[\cdot]$ is called the *hole* (also *trivial context* or $Id$, respectively), $f$ is a function symbol, $X$ is a second order variable with $ar(X) = 1$, $C$ is a context, and $t_i$ are terms. Contexts must contain exactly one occurrence of the hole. Contexts are ranged over by $C, D$. We denote contexts as $C[\cdot]$, or as $C$, if it is not ambiguous, and the subterm $X([\cdot])$ is abbreviated as $X(\cdot)$. The notation $C[t]$ means the term where the term $t$ is plugged into the hole of $C[\cdot]$. We denote syntactic equality of contexts by $\equiv$. A *ground context* is a context without occurrences of variables, i.e., it can be seen as a ground term with a single hole, where a signature with the additional constant $[\cdot]$ is used. The length of the position of the hole of a context $C$ is called *main depth* of $C$, denoted as $|C|$. The first digit of the position of the hole of a nontrivial context $C$ is denoted as *firstdpos*$(C)$. The size of terms is the number of occurrences of symbols, and the size of contexts is the number of occurrences of symbols not counting the hole. This may be denoted as *size*$(.)$.

For two contexts $C, D$, we mean by the concatenation $C \cdot D$ the context $C[D[\cdot]]$. Usually we omit the $\cdot$ and write $CD$ for $C \cdot D$ and $CD[s]$ for $C[D[s]]$. If $k$ is a natural number and $C$ is a context, $C^k$ means the $k$-fold application, i.e., $C^1 := C, C^{i+1} := CC^i$. The notation $C^k$ is used only as abstract syntax; in the concrete syntax we mean the expanded form. The context $D$ is a *prefix* of the context $C$, iff there is some context $C'$ such that $C \equiv DC'$.

The context $C$ is a *subcontext* of a term $t$, if $t \equiv D[C[t']]$ for some context $D$ and some term $t'$. The context $C$ is a *subcontext* of a context $D$, if $D \equiv D_1CD_2$ for contexts $D_1, D_2$, or if $C$ is a subcontext of a subterm of $D$.

In order to simplify index notation for certain cyclic equations, we shall use $j\ mod{*}\ n$, with

$$j\ mod{*}\ n := \begin{cases} (j\ mod\ n) & \text{if } (j\ mod\ n) \neq 0, \\ n & \text{if } (j\ mod\ n) = 0. \end{cases}$$

[2] $K$ is the constant combinator $\lambda x.\lambda y.x$.

[3] We use contexts only if all second order variables are monadic.

**Definition 2.1.** *A* second order unification problem (SOUP) is a multiset of equations $\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$, where $s_i, t_i$ are terms. A substitution $\sigma$ such that for all $i : \sigma(s_i) \equiv \sigma(t_i)$ is called a *unifier* of $\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$.

At several places in the algorithm, we will exploit the symmetry of equations by replacing equations $s \doteq t$ in $S$ by their symmetric equivalent $t \doteq s$. If this is done, we state it explicitly.

In the following we consider restricted second order problems, where an upper bound on the number of occurrences of the lambda-bound variables in the terms in codomains of unifiers is given. This does not mean that the size of the terms in the codomain of unifiers is bounded, for example $\lambda x.\underbrace{f(\ldots(f}_{k}(x)\ldots))$ has one occurrence of the bound variable $x$, but the size grows with $k$.

**Definition 2.2.** Let $S$ be a SOUP. Let there be a function $b : Var(S) \cap \mathcal{V}_2 \to \mathbb{N} \setminus \{0\}$. Then the pair $(S, b)$ is called a *bounded SOUP (BSOUP)*. A substitution $\sigma$ is a *unifier* of a BSOUP $(S, b)$, iff $\sigma$ is a unifier of $S$ and for every variable $X \in Var(S) \cap \mathcal{V}_2$ where $\sigma(X) = \lambda y_1, \ldots, y_k.t_X$, it is $\sum_{i=1,\ldots,k} occ(y_i, t_X) \leqslant b(X)$.

**Definition 2.3.** Let $(S, b)$ be a BSOUP. A second order variable $X$ is called a *Z-context variable*, iff it has arity 1 and $b(X) = 1$. If all second order variables in $Var(S) \cap \mathcal{V}_2$ are Z-context variables, then $(S, b)$ is called a Z-context unification problem (ZCUP). In this case we usually write only $S$ instead of $(S, b)$. The size of a ZCUP is the sum of the sizes of the involved terms.

In transforming a ZCUP $S$, we will sometimes say "$X$ is instantiated by $\lambda x.t$", where $t$ is a term. This is meant to replace every occurrence in $S$ of $X(s)$ by $t[s/x]$. We will also sometimes say that $x$ is instantiated by $s$, and mean that every occurrence of $x$ in the ZCUP $S$ is replaced by $s$.

## 3. Bounded second order unification

In this section we show that decidability of $Z$-context unification is equivalent to decidability of bounded second order unification. Starting with a BSOUP $(S, b)$, we will built a finite tree of simpler BSOUPs, such that every unifiable internal node has at least one unifiable child, and every leaf is a ZCUP. Hence $(S, b)$ is unifiable, iff some of the leaves of the tree is a unifiable ZCUP.

The tree is built as follows. Given a BSOUP $(S, b)$ at some node, which is not a ZCUP, obtain the children by guessing a unifier in all possible ways:

(1) For some $X$ in $(S, b)$ with $b(X) > 1$, the instance term $\lambda x_1, \ldots, x_m.t$ may have less than $b(X)$ occurrences of $x_1, \ldots, x_m$. Then one child is the BSOUP $(S', b')$, where $S = S'$, and $b'$ is $b$ except $b'(X) = b(X) - 1$.

(2) For some $X$ in $(S, b)$ with $ar(X) > 1$, the instance term $\lambda x_1, \ldots, x_m.t$ may have no occurrence of $x_i$. Then one child is the BSOUP $(S', b')$, where $S = S'$, and $X$ is replaced by $\lambda x_1, \ldots, x_m.X(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m)$.

(3) For some $X$ in $(S, b)$ with $b(X) > 1$ and $ar(X) > 1$, the instance term $\lambda x_1, \ldots, x_m.t$ can be represented as $\lambda x_1, \ldots, x_m.A[f(t_1, \ldots, t_k)]$, where $A$ is a ground context that does not contain occurrences of $x_i$, but there are at least two indices $j, j'$, such that $t_j$ and $t_{j'}$ contain occurrences

of some $x_k$. A child $(S', b')$ is obtained by guessing the function symbol $f$, which has to occur in the BSOUP, and the distribution of the number of occurrences in $t_i$, and replacing $X$ by $\lambda x_1, \ldots, x_m.X_0(f(X_1(x_1, \ldots, x_m), \ldots, f(X_k(x_1, \ldots, x_m))))$ and adapting $b'$ accordingly, such that $b'(X_0) = 1$ and $b(X) = \sum_{i=1}^{k} b'(X_i)$, where $b'(X_i)$ is positive for at least two indices. Since the guess may be $b'(X_i) = 0$ for some indices, which is prohibited by the definition, there will also be children where a $z_i$ has to be plugged in instead of $X_i(x_1, \ldots, x_m)$.

**Lemma 3.1.** *If a BSOUP $(S, b)$ is unifiable, then there is a unifier $\sigma$ of $(S, b)$ such that every function symbol of arity $\geqslant 1$ in the codomain of $\sigma$ occurs also in $S$.*

**Proof.** Given a unifier $\tau$ of $(S, b)$ which contains a function symbol $f$ with $ar(f) \geqslant 1$ not occurring in $S$, construct a unifier $\tau'$ as follows. Let $a \in \Sigma$ be a constant. Replace all subterms $f(t_1, \ldots, t_n)$ in the codomain of $\sigma$ by the constant $a$. The number of bound variables is not increased by this replacement. □

Note that a unifier of $S$ may contain constants $a$ with $ar(a) = 0$ that are not contained in $S$.

**Definition 3.2.** The following (non-deterministic) rule is used to translate a BSOUP $(S, b)$ into another BSOUP $(S', b')$.

Given a BSOUP $(S, b)$ as an input, do the following:

Select a second order variable $X$ that occurs in $S$ which is not a Z-context variable; i.e., $b(X) > 1$ or $ar(X) > 1$. Let $V_S := Var(S) \cap \mathcal{V}_2$. Select one of the following possibilities:

(1) This selection is applicable only if $b(X) > 1$.
Define $S' := S$, $b'(X) := b(X) - 1$ and $b'(Y) := b(Y)$ for second order variables $Y \in V_S \setminus \{X\}$.

(2) This selection is applicable only if $ar(X) > 1$.
Construct $(S', b')$ as follows:
Let $X'$ be a new second order variable with $ar(X') = ar(X) - 1$. Select a number $i \in \{1, \ldots, ar(X)\}$ and replace every occurrence of $X$ as follows: replace $X(t_1, \ldots, t_i, \ldots, t_n)$ by $X'(t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$. Then define $b'(X') = b(X)$, and $b'(Y) = b(Y)$ for $Y \in V_S \setminus \{X'\}$.

(3) This selection is applicable only if $b(X) > 1$ and there is a function symbol $g$ occurring in $S$ with $ar(g) \geqslant 2$.
Select a symbol $f$ in the signature, which also occurs in $S$ with $ar(f) \geqslant 2$. Let $n := ar(f)$. Select a partition $I_1 \cup I_2$ of $\{1, \ldots, n\}$ with $2 \leqslant |I_2| \leqslant b(X)$.[4] Let $z_i$ for $i \in I_1$ be new first order variables, and $X_i$ for $i \in \{0\} \cup I_2$ be new second order variables. The following conditions must hold: $ar(X_0) = 1$, for $i \in I_2$: $ar(X_i) = ar(X)$. Select a function $b'$ such that: $b'(X_0) = 1$, $b'(X_i) \geqslant 1$ for all $i \in I_2$, $\sum_{i \in I_2} b'(X_i) = b(X)$, and $b'(Y) = b(Y)$ for $Y \in V_S \setminus (\{X, X_0, \} \cup \{X_i \mid i \in I_2\})$.
Generate the new BSOUP $S'$ from $S$ by replacing all occurrences of $X$ as follows: Every subterm $X(s_1, \ldots, s_m)$ is replaced by $X_0(f(r_1, \ldots, r_n))$, where $r_i = z_i$ for $i \in I_1$ and $r_i = X_i(s_1, \ldots, s_m)$ for $i \in I_2$.

[4] The set $I_1$ represents the indices without occurrences of holes in the instantiation.

**Proposition 3.3.** *Bounded second order unification is decidable iff Z-context unification is decidable.*

**Proof.** The idea of the proof is as follows:

We show that given a BSOUP $S_0$, the tree of all non-deterministic possibilities using the rule in Definition 3.2 can be effectively computed and is finite, and that the leaves are Z-context unification problems. The input problem is unifiable iff one of the Z-context unification problems at the leaves is unifiable.

*Termination*: Every node $S$ in the computation tree is measured by a multiset $\{(b(X), ar(X)) \mid X$ occurs in $S\}$, where the pairs are ordered lexicographically, and the multiset is ordered by the induced multiset ordering (see [2,5]). It is easy to see that every step from a father to a son in the computation tree strictly reduces the measure of the node. Since the measure is well-founded, this shows termination.

Furthermore, since the number of alternatives in every step is finite, the computation tree is finite, by König's Lemma.

It is also clear that all leaves must be ZCUPs, since otherwise, it is possible to apply the rule.

*Completeness:* We show by induction on the number of steps that for every unifiable BSOUP $(S_0, b_0)$ at the root of the tree, there is a reachable and unifiable Z-context unification problem, where the unifier is restricted such that only the following symbols from the signature occur: non-constant function symbols occurring in $S$, or constant symbols from the signature.

Lemma 3.1 shows the base case: If the root $(S_0, b_0)$ is unifiable, then there is also a unifier using only constants or non-constant function symbols also occurring in $S_0$.

Let $(S, b)$ be a BSOUP at a node in the computation tree that is not a ZCUP and that is unifiable. Then there is a unifiable son of $(S, b)$ in the computation tree: Let $\sigma$ be a unifier of $(S, b)$ which only uses function symbols in $S$. Then there are several cases:

- There is a variable $Y \in Var(S) \cap \mathcal{V}_2$ with $\sigma(Y) = \lambda x_1, \ldots, x_k.t_Y$ and $\sum_{i=1,\ldots,k} occ(x_i, t_Y) < b(Y)$. Then use alternative 1 and decrease $b(Y)$.
- There is a variable $Y \in Var(S) \cap \mathcal{V}_2$ with $\sigma(Y) = \lambda x_1, \ldots, x_k.t_Y$ and there is some $j$, such that $x_j$ does not occur in $t_Y$. Then use alternative 2 and reduce the arity of $Y$.
- There is a variable $Y \in Var(S) \cap \mathcal{V}_2$ with $\sigma(Y) = \lambda x_1, \ldots, x_k.t_Y$ and $\sum_{i=1,\ldots,k} occ(x_i, t_Y) = b(Y) > 1$ and for every $i$, the variable $x_i$ occurs in $t_Y$. Then there is a position $p$ in $t_Y$, which is the greatest common prefix of all positions of variables $x_i$ for $i = 1, \ldots, k$ in $t_Y$. The function symbol of the subterm $t_{Y|p}$ must have arity $\geqslant 2$, hence it is possible to apply the third possibility. It is straightforward to construct a unifier of the son.

*Soundness:* We have to show that if there is a unifiable ZCUP at a leaf, then the BSOUP $S_0$ at the root is also unifiable. This is a straightforward check using the rule in Definition 3.2. □

## 4. Preparatory definitions

In the following sections, we give a decision procedure for the Z-context unification problem. Hence from now on, all second order variables $X$ are Z-context variables, i.e., the arity of $X$ is 1, and an instantiation for $X$ must be of the form $\lambda x.t$, where $x$ occurs at most once in $t$.

The basic idea of the decision procedure for ZCUP is the following. From properties of context unification, it follows that each unifiable ZCUP $S_0$ has minimal unifiers whose exponent of

periodicity is bounded in terms of the size of $S_0$ (see Section 4.1). Hence, it is sufficient to search for unifiers whose exponent of periodicity does not exceed this bound $E(S_0)$.

We construct a finite tree of ZCUPs with root $S_0$, such that the children of each node $S$ are smaller than the node $S$ according to a well-founded measure $\mu$. The leaves are ZCUPs that are either obviously non-unifiable, or obviously unifiable. If the root $S_0$ is unifiable, then it is guaranteed that there will be a unifiable leaf; otherwise there will be only non-unifiable leaves.

This measure $\mu$ is a lexicographic combination $\mu(S) = (\mu_1(S), \mu_2(S), \mu_3(S))$, where $\mu_1(S)$ is the number of Z-context variables of $S_0$. For the other components it is sufficient to consider the surface parts of $S$, i.e., we ignore the subterms $r$ that are strict subterms of terms $X(s')$. The component $\mu_3(S)$ is the number of occurrences of function symbols on the surface of $S$. The remaining component $\mu_2(S)$ depends on Z-cycles, i.e., sequences

$$L = \langle s_1 \doteq t_1\lfloor\mathcal{X}_2\rfloor, s_2 \doteq t_2\lfloor\mathcal{X}_3\rfloor, \ldots, s_h \doteq t_h\lfloor\mathcal{X}_1\rfloor\rangle$$

of equations $s_i \doteq t_i$ from $S$, where $\mathcal{X}_i$ is the head variable of $s_i$, i.e., either $s_i \equiv x_i$ for a first order variable, or $s_i \equiv X_i(s_i')$ for a second order variable $X_i$, and $t\lfloor\mathcal{X}\rfloor$ means that there is surface occurrence of $\mathcal{X}$ in $t$. In addition, at least one of the $\mathcal{X}_i$ has to be a Z-context variable, and the head symbol of at least one of the $t_i$ has to be a function symbol. The component $\mu_2(S)$ is $\infty$, if there are no Z-cycles, and the minimum $\psi(L)$ of another measure $\psi(.)$ for all Z-cycles $L$ of $S$, otherwise.

The measure $\psi(L)$ of a Z-cycle $L$ depends on the length $h$ of a Z-cycle $L$ and the way how $\mathcal{X}_{i+1}$ occurs on the surface of $t_i$. In particular, $L$ is path-unique, if each $\mathcal{X}_{i+1}$ occurs only once on the surface of $t_i$. If two Z-cycles $L, L'$ have equal length, and $L$ is path-unique, and $L'$ is not, then $\psi(L') < \psi(L)$.

Each ZCUP is in a certain *standard form*, of which we distinguish four types:

(1) $S$ is without Z-cycles and without a function symbol on the surface.
(2) $S$ is without Z-cycles but with a function symbol on the surface.
(3) $S$ has a $\psi$-minimal Z-cycle that is not path-unique.
(4) $S$ has a Z-cycle, and all its $\psi$-minimal Z-cycles are path-unique.

ZCUPs of the first kind of are easily seen to be unifiable. For the other types, particular reduction rules (see Section 6) will be given that turn $S$ to *fail* or to a $\mu$-simpler ZCUP $S'$ of one of the four types, its children in the tree.

To bring the system into standard form roughly means applying the decomposition rules of first order unification. One exception is that the occurs-check rule has to ignore non-surface variable occurrences. Another exception concerns the rule that given an equation $x \doteq t$, replaces occurrences of $x$ by $t$. This rule may increase the measure, hence this rule must be used in a controlled way only, which is built into the particular reduction rules.

Each of the particular reduction rules non-deterministically guesses parts of a solution (obeying the bound $E(S_0)$), i.e., any such solution can be obtained from a solution of one of the children's $S_1', \ldots, S_k'$.

In the sequel, we will present the formal development.

### *4.1. Exponent of periodicity*

A *minimal* unifier of a ZCUP $S$ is a unifier such that the sum of the sizes of the ground terms/contexts assigned to the variables in the problem is minimal with respect to all unifiers of the prob-

lem. The *exponent of periodicity* (see also [19]) of a unifier $\sigma$ of $S$ is the maximal number $n$ such that for some variable $x$ (or $X$, respectively) occurring in the problem $S$, one of the following holds:

- $\sigma(x)$ contains a nontrivial subcontext of the form $C^n$, where $C$ is a nontrivial ground context.
- $\sigma(X) \equiv \lambda z.s$, where $z$ occurs at most once in $s$, and $s$ contains a nontrivial subcontext of the form $C^n$, where $C$ is a nontrivial ground context.

Note that a unifier $\sigma'$, where $\sigma'(\mathcal{X})$ is a subterm or subcontext of $\sigma(\mathcal{X})$, always has an exponent of periodicity not greater than that of $\sigma$.

**Lemma 4.1.** *There is a positive real constant $c$, such that for every unifiable ZCUP $S$ the exponent of periodicity of a minimal unifier of $S$ is less than $E(S) := c * (2^{2.14 * size(S)})$.*

**Proof.** Let $S$ be a ZCUP and let $\sigma$ be a minimal unifier of $S$. Construct a new ZCUP $S'$ as follows. For every second order variable $X$ where $\sigma(X)$ is a constant function, let $v_X$ be a new first order variable, and construct $S'$ by replacing every occurrence $X(t)$ by $v_X$. We define the corresponding unifier $\sigma'$ of $S'$ by $\sigma'(v_X) := \sigma(X)(a)$ for the second order variables $X$, for which $\sigma(X)$ is a constant function, and $\sigma'(\mathcal{Y}) := \sigma(\mathcal{Y})$, otherwise. Here $a$ is an arbitrary constant.

It is not hard to verify that the substitution $\sigma'$ is a minimal unifier of $S'$ as a context unification problem. Application of Theorem 30 in [19] and the subsequent remark on the context unification problems that permit first order variables in [19] now show that the exponent of periodicity of $\sigma'$ is smaller than $c * (2^{2.14 * size(S')})$ for a fixed real constant $c > 0$.

Using the minimality of $\sigma$, it is easy to see that variables in $Var(S) \setminus Var(S')$ have only minimal instantiations: i.e., $a$ or $(K\ a)$ for some constant $a$. This implies that the exponent of periodicity of $\sigma$ equals the exponent of periodicity of $\sigma'$. Furthermore, the estimation $size(S') \leqslant size(S)$ holds. Hence $c * (2^{2.14 * size(S)})$ is an upper bound for the exponent of periodicity of $\sigma$. □

### 4.2. Definitions of soundness and completeness

**Definition 4.2.** A non-deterministic transformation rule or a set of transformations rules $\mathcal{T}$ that non-deterministically transforms a ZCUP $S$ into another ZCUP $S'$ is called:

- *sound*, if whenever $S$ is transformed by $\mathcal{T}$ into $S'$, and $S'$ is unifiable, then $S$ is unifiable.
- *complete for bound $E$ and for the set $\mathcal{S}$ of ZCUPs*, iff the following holds: for all $S \in \mathcal{S}$, if $S$ has a unifier with exponent of periodicity not greater than $E$, then $\mathcal{T}$ can transform $S$ into a ZCUP $S'$ that has a unifier with exponent of periodicity not greater than $E$.
- *deterministically complete (for bound $E$)*, iff the following holds: if $S$ has a unifier with exponent of periodicity not greater than $E$, then whenever $\mathcal{T}$ transforms $S$ into a ZCUP $S'$, then $S'$ has a unifier with exponent of periodicity not greater than $E$.

When the notion "(deterministically) complete for bound $E$" is used in the following sections (subsections), the upper bound $E$ is usually already fixed and known, so we may only use "(deterministically) complete".

### 4.3. Decomposition rules

In order to describe the main reduction techniques, the following notions play a central role. Note that we use $1 \leqslant i \; mod* \; n \leqslant n$.

**Definition 4.3.** A position $p$ in the term $t$ is *on the surface* of $t$, iff there is no proper prefix $p'$ of $p$ such that $t|_{p'}$ is of the form $Y(s)$. The depth of a surface position $p$ is the length of $p$.

A variable $x$ (a second order variable $X$, a function symbol $f$, respectively) *occurs on the surface* of $t$, iff $x$ (or a term $X(s)$ for some $s$, or $f(t_1, \ldots, t_n)$, for some $t_i$, respectively) occurs on a surface position of $t$.

We use the notation $t\lfloor s \rfloor_p$ (or $t\lfloor X \rfloor_p$, respectively) to indicate that $t$ has a surface occurrence of $s$ (or $X$, respectively) at position $p$. If $p$ is not relevant, then it may be omitted.

**Definition 4.4.** Let $S$ be a ZCUP. The decomposition rules are defined in Table 1. Note that since ZCUPs are multisets, union of multisets is like disjoint union of sets.

The two rules (clash) and (occurs-check) are also called *failure rules.*

A ZCUP $S$ is *decomposed* if no decomposition rule applies to it.

Note that in a decomposed ZCUP only the equations of the form $X(s) \doteq Y(t)$ cannot be oriented.

**Example 4.5.** This example should indicate some differences of unification of ZCUPs to first order unification and context unification.

Table 1
The decomposition rules

| | | |
|---|---|---|
| (decomp) | $\dfrac{\{f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)\} \cup S}{\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\} \cup S}$, | $n = 0$ is permitted |
| (repvv) | $\dfrac{\{x \doteq y\} \cup S}{S'}$, | $S'$ is constructed from $S$ by replacing all occurrences of $x$ by $y$. |
| (orient1) | $\dfrac{\{t \doteq x\} \cup S, \quad t \notin \mathcal{V}}{\{x \doteq t\} \cup S}$ | |
| (orient2) | $\dfrac{\{t \doteq X(s)\} \cup S}{\{X(s) \doteq t\} \cup S}$ | if $t$ is of the form $f(\ldots)$ |
| (clash) | $\dfrac{\{f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_m)\} \cup S}{fail}$ | if $f \not\equiv g$ |
| (occurs-check) | $\dfrac{\left\{\begin{array}{ll} x_1 & \doteq t_1\lfloor x_2 \rfloor, \\ & \vdots \\ x_{n-1} & \doteq t_{n-1}\lfloor x_n \rfloor, \\ x_n & \doteq t_n\lfloor x_1 \rfloor \end{array}\right\} \cup S}{fail}$ | if some $t_i$ is of the form $f(s_1, \ldots, s_m)$ |

The ZCUP $\{x \doteq X(x)\}$ is unifiable. A unifier is: $\{X \to K\ a, x \to a\}$. In this ZCUP it is not possible to eliminate all the occurrences of $x$ by a variable replacement using $x \doteq X(x)$.

The ZCUP $\{X(a) \doteq X(b)\}$, where $a, b$ are different constants is unifiable. Since $a \doteq b$ is not unifiable, it is not possible to use decomposition for ZCUPs as it is valid for context equations.

**Lemma 4.6.** *The decomposition rules are sound, deterministically complete and terminate.*

**Proof.** It is easy to see that soundness and deterministic completeness (for any bound $E$) hold. Nevertheless, we show deterministic completeness of the rule (occurs-check). This means to argue that if (occurs-check) is applicable to $S$, then $S$ is not unifiable. Assume that $\sigma$ is a unifier of $S$, and there is a set of equations $x_1 \doteq t_1\lfloor x_2 \rfloor, \ldots, x_n \doteq t_n\lfloor x_1 \rfloor$ in $S$. We have that for all $i$: $size(\sigma(x_i)) \geqslant size(\sigma(x_{i+1\ mod* n}))$, and that $size(\sigma(x_i)) > size(\sigma(x_{i+1\ mod* n}))$ for at least one $i$, which is a contradiction.

To show termination, define the lexicographic measure $\nu$ of $S$ with the components:

(1) The size of $S$.

(2) The number of equations that are oriented in the wrong direction, i.e., of equations which are either of the form $t \doteq x$, where $t \notin \mathcal{V}$, or of the form $f(s_1, \ldots, s_m) \doteq X(t)$.

Termination holds, since every application of a decomposition rule either results in fail, or strictly decreases this measure: (decomp) and (repvv) strictly decrease $\nu_1$, and (orient1) and (orient2) leave $\nu_1$ invariant, but strictly decrease $\nu_2$. □

### *4.4. Z-cycles*

**Definition 4.7.** Let $S$ be a ZCUP. A *Z-cycle* of $S$ is a sequence

$$s_1 \doteq t_1, \ldots, s_h \doteq t_h$$

of length $h \geqslant 1$ of equations, such that the following holds:

(1) For all $i$: $s_i \doteq t_i$ or $t_i \doteq s_i$ is in $S$.

(2) For all $1 \leqslant i \leqslant h$: either $s_i$ is a variable $x_i$ that occurs on the surface of $t_{i-1\ mod* h}$, or $s_i \equiv X_i(\ldots)$, and $X_i$ occurs on the surface of $t_{i-1\ mod* h}$.
Moreover, there should be at least one term $t_i$ of the form $f(t_{i,1}, \ldots, t_{i,n})$ and at least one term $s_i$ of the form $X_i(s_{i,1})$.

A Z-cycle is *path-unique* if for every $1 \leqslant i \leqslant h$ there is only one occurrence of $x_i$ (or $X_i$, respectively) on the surface of $t_{(i-1)\ mod* h}$.

The *length of a Z-cycle* $s_1 \doteq t_1, \ldots, s_h \doteq t_h$ is $h$, the length of the sequence. If for some Z-cycle $L$, there is no other Z-cycle of $S$ with a smaller length, then we say $L$ is a *length-minimal* Z-cycle.

A Z-cycle $s_1 \doteq t_1, \ldots, s_h \doteq t_h$ is called *compressed*, iff there is no $i$ such that $s_i$ or $t_i$ is a first order variable.

**Example 4.8.** We give some examples for Z-cycles and non-Z-cycles for appropriate $S$.

The sequence $x \doteq h(X(s_1)), X(s_2) \doteq x$ is a Z-cycle of length 2. The sequence $X_1(s_1) \doteq f(X_1(s_2), X_2(X_1(s_3)))$ is a path-unique and compressed Z-cycle of length 1. The sequence $X_1(x_1) \doteq X_2(X_1(x_1))$ is not a Z-cycle. The sequence $x_1 \doteq X_1(y_1), X_1(y_2) \doteq x_2, x_2 \doteq f(x_1)$ is a non-compressed Z-cycle.

**Definition 4.9.** Let $S$ be a ZCUP and $L$ be a Z-cycle of $S$ of the form $s_1 \doteq t_1, \ldots, s_h \doteq t_h$. For each of the terms $t_i, 1 \leqslant i \leqslant h$, let $C_i$ be the context determined as follows: Let $r_i$ be the smallest subterm of $t_i$, such that all surface occurrences of $\mathcal{X}_{(i+1 \bmod * h)}$ from $t_i$ are also contained in $r_i$. The *relevant context* $C_i$ of equation $i$ is uniquely determined by $t_i \equiv C_i[r_i]$.

Note that for a path-unique Z-cycle, the head of term $r_i$ (where $t_i \equiv C_i[r_i]$) is the variable $\mathcal{X}_{i+1 \bmod * h}$.

**Example 4.10.** Let $X_i(s) \doteq g(f(X_{i+1}(t_1), X_{i+1}(t_2)))$ be a part of a Z-cycle, then the relevant context is $C_i \equiv g([\cdot])$.

The following lemma shows that in a length-minimal Z-cycle $s_1 \doteq t_1, \ldots, s_h \doteq t_h$, the head variables of $s_i$ have only surface occurrences in the expected terms $t_{i-1 \bmod * h}$ for $i = 1, \ldots, h$. But note that there is no restriction on non-surface occurrences.

**Lemma 4.11.** *Let $S$ be an arbitrary ZCUP, and $L$ be a Z-cycle of minimal length in $S$. If $L$ is $s_1 \doteq t_1, \ldots, s_h \doteq t_h$, and $\mathcal{X}_i$ is the head variable of $s_i$, then for all $j$: if there is a surface occurrence of $\mathcal{X}_j$ in $t_k$, then $k = j - 1 \, mod * h$.*

**Proof.** If there is a surface occurrence of $\mathcal{X}_j$ in $t_k$, where $k \neq j - 1 \, mod * h$, then it is possible to extract a shorter Z-cycle from $L$. □

*4.5. A well-founded measure for termination*

**Definition 4.12.** The lexicographic measure $\psi(L) = (\psi_1(L), \psi_2(L), \psi_3(L))$ of a Z-cycle $L$ of a ZCUP $S$ has the following three components:

$\psi_1(L) =$ The length $h$ of $L$.

$\psi_2(L) =$ 0, if $L$ is non-path-unique, and 1, if $L$ is path-unique.

$\psi_3(L) =$ –If $L$ is non-path-unique, then the minimal main depth of a relevant context $C_j$ of $L$ where $t_j$ contains at least two different surface occurrences of $X_{(j+1) \bmod * h}$.
–If $L$ is path-unique, then the number of indices $1 \leqslant i \leqslant h$ such that the relevant context $C_i$ of the $i$th equation is not trivial.

The measures $\psi(L)$ and $\psi(L')$ of two different Z-cycles $L, L'$ are compared lexicographically.

**Definition 4.13.** The measure $\mu(S)$ of a ZCUP $S$ is a lexicographic one with the components $\mu_1(S), \mu_2(S), \mu_3(S)$.

$\mu_1(S) =$ The number of second order variables of $S$.

$\mu_2(S) =$ If there is a Z-cycle in $S$, then $\min\{\psi(L) \mid L \text{is a Z-cycle of } S\}$; otherwise, $\infty$.

$\mu_3(S) =$ The number of occurrences of function symbols in $S$ on surface positions.

For $i = 1, 2, 3$, we will write $\mu_{2,i}(S)$ for $\psi_i(L)$ of a $\psi$-minimal Z-cycle $L$ of $S$. If there is no Z-cycle, then the components are defined to be $\infty$.

**Lemma 4.14.** *The measure $\mu$ for ZCUPs is well-founded.*

The following example shows that it makes sense to prefer the non-path-unique Z-cycles to the path-unique ones w.r.t. the ordering, and also motivates the second component of $\psi$.

**Example 4.15.** The rule (repvv) may turn a path-unique Z-cycle into a non-path-unique one Let

$$S := \{x \doteq f(y_1, y_2), y_1 \doteq X(y_3), X(y_4) \doteq x, y_2 \doteq y_1\}.$$

The sequence $x \doteq f(y_1, y_2), y_1 \doteq X(y_3), X(y_4) \doteq x$ is a path-unique Z-cycle. An application of (repvv) using the replacement $y_2 \to y_1$ turns this into the non-path-unique Z-cycle $x \doteq f(y_1, y_1), y_1 \doteq X(y_3), X(y_4) \doteq x$.

### *4.6. Properties of decomposition*

**Lemma 4.16.** *Let $S$ be an arbitrary ZCUP, and let $S'$ be the result after several applications of decomposition rules to $S$. If there is no fail, then the following holds*:
(1) *The number of second order variables in $S$ and $S'$ is the same.*
(2) $\mu(S) \geqslant \mu(S')$.

**Proof.** It is sufficient to consider one application of a decomposition rule.

There is no decomposition rule that can modify the number of second order variables, hence the first claim holds.

There are several cases:

- If $S, S'$ do not contain a Z-cycle, then the number of function symbols on the surface cannot increase, and since $\mu_2(S) = \mu_2(S') = \infty$, we have $\mu(S) \geqslant \mu(S')$.
- If $S$ does not contain a Z-cycle, but $S'$ does, then $\mu_2(S) > \mu_2(S')$, hence $\mu(S) \geqslant \mu(S')$.
- If $S$ contains a Z-cycle, then this also holds for $S'$: Only (repvv) may have an influence on a Z-cycle. If there is a Z-cycle $L$ which contains two equations $s_j \doteq t_j\lfloor x_{j+1}\rfloor, x_{j+1} \doteq t_{j+1}$, then a replacement of $x_{j+1}$ by $y$ turns $L$ into a Z-cycle $L'$. This also holds, if there are two equations $s_j \doteq t_j\lfloor x_{j+1}\rfloor, x_{j+1} \doteq x_{j+2}$, and $x_{j+1}$ is replaced by $x_{j+2}$. This reasoning also shows that the length of the Z-cycle is not increased by (repvv). Application of (decomp) may introduce Z-cycles, but not increase the length of existing ones. Hence $\mu_{2,1}(S) \geqslant \mu_{2,1}(S')$.

The same reasoning also shows that (repvv) cannot turn a non-path-unique Z-cycle into a path-unique one, hence $\mu_{2,2}(S) \geqslant \mu_{2,2}(S')$.

Now we argue that the main depths of the relevant contexts in a minimal-length Z-cycle can only be decreased by (repvv): Let an equation $s_i \doteq C_i\lfloor r_i\rfloor, z \doteq t_{i+1}$ be in a Z-cycle, and assume that $\mathcal{X}_{i+1} \doteq z$. If the replacement is $y \to z$, such that $C_i\lfloor r_i\rfloor[y/z] \equiv s'_i \equiv C'_i\lfloor r'_i\rfloor$, then there may be more occurrences of $z$ in $s_i$, hence $C'_i$ is a prefix of $C_i$, and the main depth of $C'_i$ is not greater than the main depth of $C_i$. In the case that the replacement is $z \to y$, the reasoning is analogous to the previous case. This works in the case where $y \not\equiv t_{i+1}$ as well as in the case $y \equiv t_{i+1}$.

Hence $\mu_{2,3}(S) \geqslant \mu_{2,3}(S')$. Concluding, we have shown $\mu(S) \geqslant \mu(S')$. □

*4.7. Replacement of first order variables*

This paragraph discusses the rule (repvt) which replaces surface occurrences of a variable $x$ by the term $t$, if the equation $x \doteq t$ is in the ZCUP. The examples in this paragraph show that (repvt) may increase the measure $\mu$ of ZCUPs. The application of this rule is necessary under certain circumstances in the rules in Section 6, however, there will always be side-conditions that prevent the increase of the measure $\mu$

$$\text{(repvt)}\ \frac{\{x \doteq t\} \cup S, \quad t \notin \mathcal{V}}{\{x \doteq t\} \cup S'},$$

$S'$ is constructed from $S$ by replacing all surface occurrences of $x$ by $t$. It is only applicable, if $t$ has no surface occurrence of $x$.

We also sometimes say the rule (repvt) is applied for the equation $x \doteq t$.

**Lemma 4.17.** *The rule (repvt) is sound and deterministically complete.*

**Proof.** If (repvt) is applied for $x \doteq t$ transforming $S$ into $S'$, then the claim of the lemma follows from the fact, that the equation $x \doteq t$ is not removed, and hence for every unifier $\sigma$ of $S$ and $S'$, the equation $\sigma(x) = \sigma(t)$ holds. □

**Example 4.18.** An application of (repvt) may transform a ZCUP with a Z-cycle into a ZCUP without Z-cycles.

The ZCUP $\{x \doteq f(y), \quad y \doteq X(s_1), \quad X(s_2) \doteq f(x), \quad x \doteq f(a)\}$ has a nontrivial Z-cycle. After instantiating it with $\{x \to f(a)\}$, there is no Z-cycle anymore: $\{f(a) \doteq f(y), \quad y \doteq X(s_1), \quad X(s_2) \doteq f(f(a)), \quad x \doteq f(a)\}$.

**Example 4.19.** Using (repvt) for an equation $x \doteq t$ that is not contained in a length-minimal Z-cycle may increase the minimal length of Z-cycles: Let

$$S := \left\{ \begin{array}{ll} x & \doteq f(a, X_1(\ldots)), \\ X_1(\ldots) & \doteq f(a, x), \\ x & \doteq f(Y_1(\ldots), b), \\ Y_1(\ldots) & \doteq Y_2(\ldots), \\ Y_2(\ldots) & \doteq Y_3(\ldots), \\ Y_3(\ldots) & \doteq f(x, b). \end{array} \right\}$$

Applying (repvt) for $x \doteq f(Y_1(\ldots), b)$ modifies the equations, such that the first two equations no longer form a Z-cycle, even after using (decomp). After the replacement, the length-minimal Z-cycle is of length 3, whereas the length of the length-minimal Z-cycle before the replacement was 2:

$$\left\{\begin{array}{ll} Y_1(\ldots) & \doteq a, \\ b & \doteq X_1(\ldots), \\ X_1(\ldots) & \doteq f(a,x), \\ x & \doteq f(Y_1(\ldots),b), \\ Y_1(\ldots) & \doteq Y_2(\ldots), \\ Y_2(\ldots) & \doteq Y_3(\ldots), \\ Y_3(\ldots) & \doteq f(x,b). \end{array}\right\}$$

**Definition 4.20** (*Compressing Z-cycles*)**.** The following rule is a variant of (repvt) and is intended to compress length-minimal Z-cycles by replacing first order variables:

$$\text{(compress)}\ \frac{\{x \doteq t\} \cup S, \quad t \notin \mathcal{V}}{\{x \doteq t\} \cup S'}$$

if there is a length-minimal Z-cycle in the ZCUP containing $x \doteq t$, where $S'$ is constructed from S by replacing all surface occurrences of $x$ by $t$. It is only applicable, if $t$ has no surface occurrence of $x$.

*4.8. Standardization*

**Definition 4.21.** A ZCUP $S$ is called *flat*, iff the depth of all surface positions is $\leqslant 1$, and only first order variables occur at surface positions of depth 1; i.e., all terms in $S$ are of the following forms: $x, f(x_1,\ldots,x_n), X(t)$.[5]

The method for transforming a given ZCUP $S$ depends on the question whether $S$ contains a Z-cycle or not. For ZCUPs without $Z$-cycles, the termination arguments become more transparent if we move to flat ZCUPs. For ZCUPs with $Z$-cycles it is more convenient if minimal-length Z-cycles are compressed.

The rule (flatten) is intended to be applied to ZCUPs without Z-cycles.

**Definition 4.22** (*flatten*)**.**

$$\text{(flatten)}\ \frac{\{s_1 \doteq s_2\lfloor t\rfloor_p\} \cup S, \quad t \notin \mathcal{V}}{\{s_1 \doteq s_2\lfloor x\rfloor_p, x \doteq t\} \cup S},$$

where $x$ is a fresh variable and $p$ is a surface position of depth $\geqslant 1$.

Note that this covers all possibilities, since all nontrivial surface positions in a decomposed ZCUP are in the terms of the right-hand side of equations, which are of the form $f(\ldots)$.

**Lemma 4.23.** *The rule* (*flatten*) *is sound and deterministically complete and terminates.*

**Proof.** Soundness and deterministic completeness are easy.

[5] This includes constants, since we allow $n = 0$ in $f(x_1,\ldots,x_n)$.

Flattening can be applied at most as often as there are surface positions that are not at top level and the term at this position is not a first order variable. □

**Lemma 4.24.**
(1) *If* (*flatten*) *is applied to a ZCUP S without Z-cycles, then the resulting ZCUP S′ is also without Z-cycles.*
(2) *If* (*flatten*) *is applied to a decomposed ZCUP S without Z-cycles, then the resulting ZCUP S′ is also decomposed* (*and without Z-cycles*).

**Proof.** 1. If there is a Z-cycle $L$ in $S'$, but no Z-cycle in $S$, then this Z-cycle $L$ must contain the two resulting equations of (flatten) and thus can be written $s_1 \doteq t_1\lfloor x \rfloor_p, x \doteq t_2, s_3 \doteq t_3, \ldots$, since $x$ is fresh. But then the sequence of equations $s_1 \doteq t_1\lfloor t_2 \rfloor_p, s_3 \doteq t_3, \ldots$, is a Z-cycle in $S$.

2. The rule (flatten) does not introduce equations $x \doteq y$, hence (repvv) is not applicable to $S'$. It does also not introduce equations of the form $f(\ldots) \doteq f(\ldots)$, hence (decomp) is not applicable to $S'$. It is also clear that by construction, there is no orientation necessary. It is also not possible that the failure rules become applicable. □

**Definition 4.25** (*Standardization*)**.** A ZCUP is standardized as follows:
(1) First use the decomposition rules exhaustively.
(2) Then:
  (a) If the ZCUP does not have Z-cycles, then use (flatten) exhaustively.
  (b) If the ZCUP contains Z-cycles, then apply (compress) and decomposition exhaustively.

We prove some required properties of standardization.

**Lemma 4.26.**
(1) *Standardized ZCUPs are also decomposed.*
(2) *A standardized ZCUP is either flat and does not contain Z-cycles, or it contains Z-cycles, and every length-minimal Z-cycle is compressed.*

**Proof.** Since flattening in decomposed ZCUPs does not enable the application of decomposition rules, part 1 holds for ZCUPs without Z-cycles.

For ZCUPs with Z-cycles, this is obvious, since decomposition has to be applied exhaustively. Hence part 1 holds.

If a ZCUP is standardized, and there are no Z-cycles, then this is the case where (flatten) was applied exhaustively.

If a ZCUP is standardized, and there are Z-cycles, then the rule (compress) was applied exhaustively. There are no uncompressed, length-minimal Z-cycles, since otherwise (compress) would be applicable. □

**Lemma 4.27.** *Standardization is sound, deterministically complete and terminates.*

**Proof.** Soundness and deterministic completeness are obvious, since this holds for decomposition by Lemmas 4.6, 4.23 and 4.17, and since (compress) is a special case of (repvt)

To show termination, we first use Lemma 4.6 to show that decomposition terminates. Now we distinguish two cases:

If there is no Z-cycle after decomposition, then flattening terminates using Lemma 4.23.

If there is a Z-cycle after decomposition, then every application of (compress) reduces the minimal length of Z-cycles at least by 1; moreover, decomposition does not increase the minimal length of Z-cycles (see Lemma 4.16). □

**Lemma 4.28.** *Let $S$ be an arbitrary ZCUP, and let $S'$ be the result of standardizing $S$. If there is no fail, then the following holds:*
(1) *The number of second order variables in $S$ and $S'$ is the same.*
(2) $\mu(S) \geqslant \mu(S')$.

**Proof.** There is no decomposition rule that can modify the number of second order variables, hence (1) holds.

To prove (2), there are several cases:

- If $S$ does not contain a Z-cycle, but decomposition introduces a Z-cycle, then $\infty = \mu_2(S) > \mu_2(S')$, hence $\mu(S) \geqslant \mu(S')$.
- If neither $S$ nor $S'$ contains a Z-cycle, then $\infty = \mu_2(S) = \mu_2(S')$, and $\mu_3(S) \geqslant \mu_3(S')$, since decomposition does not increase the number of surface function symbols. Hence $\mu(S) \geqslant \mu(S')$.
- If $S$ contains a Z-cycle, then $S'$ contains a Z-cycle. Decomposition does not increase the minimal length of Z-cycles. The rule (compress) strictly decreases the minimal length of Z-cycles, hence $\mu_{2,1}(S) \geqslant \mu_{2,1}(S')$, and $\mu_{2,1}(S) > \mu_{2,1}(S')$ if there is at least one application of (compress).
  If there is no application of (compress), then Lemma 4.16 shows the claim. □

### *4.9. Four types of unification problems*

We now introduce the following four types of ZCUPs.

**Definition 4.29.** A ZCUP $S$ is of
- type 0 if $S$ is standardized, does not have any Z-cycles, and if there is no function symbol $f$ on the surface of $S$, (also called pre-unified in the literature on higher order unification);[6]
- type 1 if $S$ is standardized, does not have any Z-cycles, and if there exists a function symbol $f$ on the surface of $S$;
- type 2 if $S$ is standardized, contains a $Z$-cycle and if there is a $\psi$-minimal $Z$-cycle that is non-path-unique;
- type 3 if $S$ is standardized, contains a $Z$-cycle and if all $\psi$-minimal $Z$-cycles are path-unique.

Note that ZCUPs of types 0 and 1 are flat, i.e. all terms $s, t$ occurring in equations $s \doteq t$ are of the form $x$, $f(x_1, \ldots, x_n)$ or $X(r)$.

Note also that in ZCUPs of types 2 and 3, every $\psi$-minimal $Z$-cycle is compressed.

[6] Since constant symbols count as function symbols, this includes that there is no constant symbol on the surface of $S$.

**Lemma 4.30.** *ZCUPs of type* 0 *are always unifiable*.

**Proof.** Let $a$ be a constant from the signature. A substitution $\sigma$ that replaces every second order variable $X$ by $K\ a$ and every first order variable $x$ by $a$ is a unifier, since every term in an equation is either a first order variable, or a term of the form $X(t)$. □

**Lemma 4.31.** *After standardization, a resulting ZCUP S has one of the types* 0–3.

## 5. Decidability of Z-context unification: the algorithm UZ

The idea behind *UZ* is as follows: given a ZCUP $S_0$ as an input, it searches for a unifier of $S$ where the exponent of periodicity does not exceed the bound $E(S_0)$. The search proceeds by transforming ZCUPs into further ZCUPs until the result is of type 0. The branching factor of every transformation is finite. It will be shown that the computation tree of the search is finite by showing that there is no infinite (non-deterministic) transformation sequence.

It may happen that some ZCUP in the computation tree has only unifiers with exponent of periodicity greater than $E$. This is neither an error nor a problem. In our algorithm it means that this branch will be redundant. It can be viewed as a missing optimization.

**Definition 5.1.** *UZ* is a non-deterministic algorithm working as follows:

Let the input be the ZCUP $S_0$.

(1) Fix the constant $E := c * (2^{2.14 * size(S_0)})$ (see Lemma 4.1).
(2) The first action is to standardize the ZCUP $S_0$. This gives a ZCUP $S_1$ of some types 0–3.
(3) Given an intermediate ZCUP $S$, the rules defined in Sections 6.1–6.3 are then used repeatedly using the bound $E$ as follows:
   - If the ZCUP $S$ has type 0, then stop with success.
   - If the ZCUP $S$ has type 1, then apply the rule (Imitation) in Section 6.1.
   - If the ZCUP $S$ has type 2, then apply the rule (solve-ambiguous-Z-cycle) in Section 6.2.
   - If the ZCUP $S$ has type 3 and $\mu_{2,3} \geqslant 2$, then apply the rule (shuffle*) in Section 6.3.
   - If the ZCUP $S$ has type 3 and $\mu_{2,3} = 1$, then apply the rules (solve-unique-Z-cycle) in Section 6.3.

The repetitions may either stop with Fail, or with success if a ZCUP of type 0 is reached.

The ZCUP $S_0$ is recognized as unifiable, iff there is an execution possibility of UZ that stops with success.

We give an example for the run of the algorithm.

**Example 5.2.** The following example should illustrate the ideas of the algorithm UZ, though not all possibilities are covered. Examples for more complex situations are in Section 6.

Let $S_0 := \{X(y_1) \doteq f(x_1, x_2), f(g(x_1), x_3) \doteq X(x_1)\}$.

Lemma 4.1 allows us to fix a bound $e$ on the exponent of periodicity.

The ZCUP has no Z-cycle. After standardization, we have

$$\{X(y_1) \doteq f(x_1,x_2), X(x_1) \doteq f(x_4,x_3), x_4 \doteq g(x_1)\}.$$

In this case the ZCUP is of type 1, and an instantiation of $X$ will make progress. Of course, there are several possibilities.

(1) $X$ may be the identity. In this case the second and third equations result in $x_1 \doteq f(x_4,x_3), x_4 \doteq g(x_1)$, which would result in a failure due to occurs-check.

(2) $X$ may be a constant function, say $K\,z$. In this case the ZCUP after instantiation is

$$\{z \doteq f(x_1,x_2), z \doteq f(x_4,x_3), x_4 \doteq g(x_1)\}.$$

Now the rules of UZ are able to show that this first order unification problem is not unifiable. It is of type 1. The next operations are instantiating $z$ by $f(z_1,z_2)$, then applying (decomp) two times, then several applications of (repvv), and an application of (orient1). Finally, the obtained ZCUP is

$$\{x_1 \doteq g(x_1)\}.$$

Now there is a failure due to occurs-check.

(3) $X$ may be instantiated by $f(X'(\cdot), y_2)$. This results in

$$\{f(X'(y_1), y_2) \doteq f(x_1,x_2), f(X'(x_1), y_2) \doteq f(x_4,x_3), x_4 \doteq g(x_1)\}.$$

Application of (decomp), two applications of (repvv), and applications of (orient1) and (orient2) result in

$$\{x_1 = X'(y_1), x_4 = X'(x_1), x_4 \doteq g(x_1)\}.$$

Now there is a Z-cycle. Using (compress) results in

$$\{x_1 \doteq X'(y_1), x_4 = X'(x_1), X'(x_1) \doteq g(X'(y_1))\}.$$

Now the ZCUP has a compressed Z-cycle.
One possibility to proceed is to select an exponent, say $10 \leqslant E$ and to instantiate $X'$ by $g^{10}(\cdot)$. This generates the following ZCUP:

$$\{x_1 \doteq g^{10}(y_1), x_4 = g^{10}(x_1), g^{10}(x_1) \doteq g(g^{10}(y_1))\}.$$

It is not hard to see that after some applications of decomposition rules the occurs-check reports failure.
A successful possibility is to select 1 as exponent and to instantiate $X'$ by $g(\cdot)$. This generates the following ZCUP:

$$\{x_1 \doteq g(y_1), x_4 = g(x_1), g(x_1) \doteq g(g(y_1))\}.$$

After (decomp), the result is

$$\{x_1 \doteq g(y_1), x_4 = g(x_1), x_1 \doteq g(y_1)\}.$$

Then $x_4$ is instantiated by $g(y_2)$ followed by decomposition and (repvv), after this, $x_1$ is instantiated by $g(y_3)$ followed by decomposition and (repvv), which leaves an empty ZCUP, hence success is reported.

The rest of the paper is devoted to constructing the rules and proving their properties.

We prove that UZ is a decision algorithm based on the lemmas in Section 6 which state that the rules for every type of ZCUPs are sound and complete for the respective type of ZCUPs, and reduce the measure $\mu$.

**Theorem 5.3.** *UZ is a decision algorithm for ZCUPs.*

**Proof.** Given an initial ZCUP $S_0$, let $E$ be the upper bound as defined in the algorithm *UZ*. First, Lemma 4.26 shows that the initial step of *UZ* is sound and deterministically complete. The Lemmas 6.2, 6.4, 6.10, and 6.12 show all the required facts:

The transformations terminate, since every rule application strictly reduces $\mu$, and since $\mu$ is well-founded (see Lemma 4.14). Together with the fact that every rule has only a finite number of possibilities (i.e. branching is finite), König's Lemma implies that the computation tree is finite.

It is also clear that the only possibilities for the leaves in the computation tree is a failure or a ZCUP of type 0. ZCUPs of type 0 are unifiable (Lemma 4.30).

Now let $S_0$ be unifiable. Lemma 4.1 shows that there is a minimal unifier with exponent of periodicity not exceeding $E$ as given above. Completeness of the rules for the respective types 1, 2, and 3 for bound $E$ and finiteness of the computation tree show that there is a unifiable leaf, which must be of type 0.

Let $S_0$ be not unifiable. Then all leaves must be failure leaves, since otherwise there is a leaf of type 0, and then soundness of the rules would imply that $S_0$ is unifiable. □

**Theorem 5.4.** *Unifiability of ZCUPs is decidable.*

**Proof.** Follows from Theorem 5.3, since UZ is a decision procedure for unifiability of ZCUPs. □

**Theorem 5.5.** *Bounded second order unification is decidable.*

**Proof.** This follows from Theorem 5.4 and Proposition 3.3 □

## 6. Main reduction rules

In this section we describe the reduction of ZCUPs of types 1–3.

### *6.1. Reduction of ZCUPs of type 1*

The rule for transforming ZCUPs of type 1 is intended to guess the instantiation of variables in a top down fashion, such that after decomposition, at least one function symbol is removed.

Since the ZCUP is flat and decomposed, and hence there is no occurs-check failure, we will find variables on surface position, for which all surface positions have depth 0. A simultaneous instantiation and subsequent decomposition of such a set of variables is the main operation of the following rule.

Let $S$ denote a ZCUP of type 1, with a set of variables $\mathcal{V}_S := Var(S)$. Let the relations "$\sim_1$" and "$>_1$" on $\mathcal{V}_S$ be defined as follows: If $x \doteq Y(s) \in S$, then $x \sim_1 Y$; if $X(s) \doteq Y(t) \in S$, then $X \sim_1 Y$. If $x \doteq f(y_1, \ldots, y_n) \in S$, then $x >_1 y_i$ for $i = 1, \ldots, n$, and if $X(s) \doteq f(y_1, \ldots, y_n) \in S$, then $X >_1 y_i$ for $i = 1, \ldots, n$.

Let "$\sim$" denote the equivalence relation in $\mathcal{V}_S$ generated by "$\sim_1$". Denote the equivalence class of a variable $\mathcal{X}$ by $[\mathcal{X}]_\sim$. For equivalence classes $D_1, D_2$ of $\mathcal{V}_S/\sim$ define $D_1 \rhd_1 D_2$ if there exist $\mathcal{X}_i \in D_i$ such that $\mathcal{X}_1 >_1 \mathcal{X}_2$. Let "$\rhd$" denote the transitive closure of "$\rhd_1$". Note that the relation "$\rhd$" is an irreflexive partial order on $\mathcal{V}_S/\sim$, if the ZCUP is of type 1, since the ZCUP $S$ does not contain a Z-cycle and is decomposed.

**Definition 6.1** (*Imitation*)**.** Let $S$ be a ZCUP of type 1. Select a $\rhd$-maximal $\sim$-equivalence class $D$ and a function symbol $f$ according to the following conditions: there must be an equation $x \doteq f(\ldots)$ or $X(s) \doteq f(\ldots)$ in $S$, where $x \in D$ (or $X \in D$, respectively). Let $n := ar(f)$.

Then select one of the following possibilities:

(1) Select a second order variable $X' \in D$ and instantiate $X'$ by $Id$ or by $K\ y$ for some new first order variable $y$. Then standardize the ZCUP.

(2) (a) For every variable $y \in D$ instantiate $y$ by $f(y_1, \ldots, y_n)$, where $y_i$ are new variables. For every second order variable $Y \in D$ select some index $i$ and instantiate $Y$ by $f(y_1, \ldots, y_{i-1}, Y'(\cdot), y_{i+1}, \ldots, y_n)$, where $y_j$ and $Y'$ are new.
(b) Standardize the ZCUP.

**Lemma 6.2.** *The rule* (*imitation*) *is sound and complete for ZCUPs of type* 1. *If $S$ is a ZCUP of type* 1*, then application of* (*imitation*) *either fails or results in a ZCUP $S'$ of type* 0, 1, 2 *or* 3*, such that* $\mu(S') < \mu(S)$.

**Proof.** A $\rhd$-maximal equivalence class with the required properties exists, since there are no Z-cycles, there is no occurs-check failure since $S$ is decomposed, and the ZCUP is not of type 0.

After step 2, the ZCUP is standardized, since in the case that there is no Z-cycle, the ZCUP is flat, and decomposition rules are not applicable.

To show soundness is standard.

For completeness note that the exponent of periodicity of a given unifier is not increased after adapting it to the new ZCUP.

We show termination: The measure $\mu$ is properly decreased if a second order variable is eliminated in selection 1. In the second case of the rule, after instantiation and decomposition, either the number of second order variables is strictly reduced, or their number is the same, and the number of surface occurrences of the function symbol $f$ is strictly reduced. This follows, since $S$ is flat and the equivalence class $D$ is maximal. In this case, all the occurrences of function symbols, which are introduced by the instantiations in step (2a), are immediately removed by decomposition in step (2b). Moreover, at least one additional occurrence of a function symbol will be removed. If a Z-cycle is introduced by the first decomposition of standardization, then $\mu$ is strictly decreased, since $\mu_2$ is strictly decreased (see also Lemma 4.28). □

### 6.2. Reduction of ZCUPs of type 2

The transformation rule in this subsection is intended to transform ZCUPs $S$ of type 2. The ZCUP $S$ is decomposed and thus has a $\psi$-minimal, compressed Z-cycle $L$ that is non-path-unique. The transformation is focussed on this Z-cycle, and in particular on a witnessing equation $X_j(s_j) \doteq C_j[t_j]$, where $t_j$ has more than one occurrence of $X_{j+1}$, and in addition the main depth of the relevant context $C_j$ is minimal. Progress can be made by guessing the top level of $\sigma(X_j)$ for a unifier $\sigma$ of $S$. After instantiating $X_j$, and controlled decomposition, either the new Z-cycle has the same length, and the relevant context $C'_j$ has a smaller main depth, or there is a new Z-cycle that is shorter.

Now we prepare the formal definition of the transformation.

Let $L$ be a $\psi$-minimal, compressed Z-cycle of $S$ that is non-path-unique. We assume that $L \subseteq S$, by orienting equations of the form $X(s) \doteq Y(t)$, if necessary. This operation on the ZCUP is easily seen to be sound, deterministically complete, and without consequences for the measure.

We may assume that $L$ has the form $X_1(s_1) \doteq t_1, \ldots, X_h(s_h) \doteq t_h$. The Z-cycle could as well be represented as $X_1(s_1) \doteq C_1[t'_1], \ldots, X_h(s_h) \doteq C_h[t'_h]$, where $C_i$ for $i = 1, \ldots, h$ are the relevant contexts (see Definition 4.9).

**Definition 6.3** (*solve-ambiguous-Z-cycle*)**.** The input is the ZCUP $S$ of type 2 with a $\psi$-minimal, compressed Z-cycle $L$ as described above. Select one of the following possibilities:

(1) Select one of the variables $X_i$, for some $1 \leqslant i \leqslant h$, and instantiate $X_i$ either with $Id$ or with $K\ x$ where $x$ is new. Then standardize the resulting ZCUP.
(2) Select an index $j$ such that $X_j(s_j) \doteq t_j$ is an equation in $L$ such that $X_{(j+1 \bmod h)}$ occurs at least twice on the surface of $t_j \equiv f(t_{j,1}, \ldots, t_{j,m})$ and the main depth of the relevant context $C_j$ is minimal in $L$ under this condition. Now apply the following steps:
   (a) Select an index $r \in \{1, \ldots, m\}$. In the special situation where $h = 1$, the selection of $r$ is subject to the following condition: all surface occurrences of $X_1$ in $f(t_{1,1}, \ldots, t_{1,m})$ have to be in $t_{1,r}$. If this is not possible since $C_1$ is trivial, then stop with fail.
   (b) Instantiate $X_j$ by $f(x_1, \ldots, x_{r-1}, X'_j(\cdot), x_{r+1}, \ldots, x_m)$, where $x_i$ and $X'_j$ are fresh.
   (c) Apply rule (decomp) to the equation that is obtained from the equation $X_j(s_j) \doteq t_j$ by step (2b).
   (d) Apply (repvv) or (repvt) for all the new equations $x_i \doteq t_{j,i}$ $(1 \leqslant i \leqslant m, i \neq r)$ that are obtained from the previous step.
   (e) Then standardize the resulting ZCUP.

The explicit control in steps (2c) and (2d) is necessary to assure in the case where there is a Z-cycle of length $> 1$, that after application of the rule, there is a shorter and compressed Z-cycle.

**Lemma 6.4.** *The rule* (*solve-ambiguous-Z-cycle*) *is sound and complete for ZCUPs of type* 2.

*Let $S$ be a ZCUP of type* 2*, then application of* (*solve-ambiguous-Z-cycle*) *either fails, or the output is a ZCUP $S'$ of types* 0–3*, and $\mu(S') < \mu(S)$.*

**Proof.** See Appendix A.1. □

**Example 6.5.** We give several examples for the application of the rule (solve-ambiguous-Z-cycle) to a ZCUP with a non-path-unique Z-cycle.

(1) A shorter Z-cycle will be immediately generated. Let

$$S := \left\{ \begin{array}{l} X_1(s_1) \doteq h(X_2(s_2)), \\ X_2(s_3) \doteq f(X_3(s_4), X_3(s_5), s_6), \\ X_3(s_7) \doteq X_1(s_8). \end{array} \right\}$$

Application of (2b) with $j = 2, r = 3$ using the instantiation $X_2 \to f(y_1, y_2, X_2'(\cdot))$ results in:

$$\left\{ \begin{array}{ll} X_1(s_1) & \doteq h(f(y_1, y_2, X_2'(s_2))), \\ f(y_1, y_2, X_2'(s_3)) & \doteq f(X_3(s_4), X_3(s_5), s_6), \\ X_3(s_7) & \doteq X_1(s_8). \end{array} \right\}$$

Decomposition according to (2c) results in

$$\left\{ \begin{array}{ll} X_1(s_1) & \doteq h(f(y_1, y_2, X_2'(s_2))), \\ y_1 & \doteq X_3(s_4), \\ y_2 & \doteq X_3(s_5), \\ X_2'(s_3) & \doteq s_6, \\ X_3(s_7) & \doteq X_1(s_8). \end{array} \right\}$$

Now there are at least two new Z-cycles: One via $y_1$, the other via $y_2$. Applying (repvt) for the equations $y_1 \doteq X_3(s_4), \quad y_2 \doteq X_3(s_5)$ as described in (2d) results in

$$\left\{ \begin{array}{ll} X_1(s_1) & \doteq h(f(X_3(s_4), X_3(s_5), X_2'(s_2))), \\ X_3(s_7) & \doteq X_1(s_8), \\ y_1 & \doteq X_3(s_4), \\ y_2 & \doteq X_3(s_5), \\ X_2'(s_3) & \doteq s_6, \end{array} \right\}$$

and there is a shorter compressed Z-cycle.

(2) The main depth of a relevant context is decreased: Let

$$S := \left\{ \begin{array}{l} X_1(s_1) \doteq h(X_2(s_2)), \\ X_2(s_3) \doteq h(f(X_3(s_4), X_3(s_5))), \\ X_3(s_6) \doteq X_1(s_7). \end{array} \right\}$$

The relevant context $C_2 \equiv h([\cdot])$ has main depth 1. Application using the instantiation $X_2 \to h(X_2'(\cdot))$ results in:

$$\left\{ \begin{array}{ll} X_1(s_1) & \doteq h(h(X_2'(s_2))), \\ h(X_2'(s_3)) & \doteq h(f(X_3(s_4), X_3(s_5))), \\ X_3(s_6) & \doteq X_1(s_7). \end{array} \right\}$$

Decomposition results in

$$\left\{ \begin{array}{ll} X_1(s_1) & \doteq h(h(X_2'(s_2))), \\ X_2'(s_3) & \doteq f(X_3(s_4), X_3(s_5)), \\ X_3(s_6) & \doteq X_1(s_7). \end{array} \right\}$$

The relevant context $C_2' \equiv [\cdot]$ of the second equation now has depth 0.

(3) The Z-cycle has length 1.
Let $S = \{X_1(s_1) \doteq h(f(X_1(s_2), X_1(s_3)))\}$.
The first application of (solve-ambiguous-Z-cycle) using $X_1 \to h(X_1'(\cdot))$ results after decomposition in:

$$\{X_1'(s_1) \doteq f(h(X_1'(s_2)), h(X_1'(s_3)))\}.$$

The ZCUP is of type 2, hence the rule (solve-ambiguous-Z-cycle) has to be applied. If selection (2) is chosen, then there will be a fail in step (2a).
Suppose, we applied the instantiation in step (2b) with $r = 1$: an application of $X_1' \to f(y_1, X_1''(\cdot))$ results in

$$\{f(y_1, X_1''(s_1)) \doteq f(h(f(y_1, X_1''(s_2))), \quad h(f(y_1, X_1''(s_3))))\}.$$

Application of decomposition results in:

$$\left\{ \begin{array}{ll} y_1 & \doteq h(f(y_1, X_1''(s_2))), \\ X_1''(s_1) & \doteq h(f(y_1, X_1''(s_3))). \end{array} \right\}$$

Now the first equation enforces a fail due to the occurs-check.

### 6.3. Reduction of ZCUPs of type 3

Now we consider the case where $S$ is decomposed, contains a Z-cycle, and where each $\psi$-minimal Z-cycle is compressed and path-unique. The transformation is performed in two major steps.

The first step is intended to focus on a $\psi$-minimal Z-cycle $L$, and to shuffle all the relevant contexts to one index. This is done by a transformation (shuffle) that shuffles only one level of the relevant context to a neighboring index. Since such a single step may increase the measure $\mu$, in particular $\mu_{2,3}$, we define a macro step (shuffle*) that iteratedly shuffles a complete relevant context to a neighboring index until the relevant contexts of two indices are merged. This rule does not increase $\mu$, and several applications of (shuffle*) are necessary to concentrate all relevant contexts on one index of the Z-cycle.

The second step is the rule (solve-unique-Z-cycle), which guesses an instantiation along the Z-cycle. This instantiation corresponds to cycling several times along the Z-cycle, which means to instantiate by an iterated context. Here is the only place where the exponent of periodicity is used to stop instantiations.

This description is not the whole truth, since all possibilities in guessing one level of an instantiation have to be covered. An important case is that the instantiation follows the path of the Z-cycle for several levels, and then deviates. Fortunately, it can be shown that all these deviations strictly decrease $\mu$, after standardization.

Let $L$ denote a $\psi$-minimal Z-cycle of $S$. We assume that $L \subseteq S$, by orienting equations of the form $X(s) \doteq Y(t)$, if necessary. This operation on the ZCUP is easily seen to be sound, deterministically complete, and without consequences for the measure.

$L$ can be represented in the form $X_1(s_1) \doteq C_1[X_2(t_1)], \ldots, X_h(s_h) \doteq C_h[X_1(t_h)]$. We first describe the reduction in the situation where $L$ contains at least two nontrivial relevant contexts $C_j$ and $C_{j'}$. In particular we have $h \geqslant 2$. We consider the following sub-rule (shuffle), which can be applied to a ZCUP only through the rule (shuffle*).

**Definition 6.6** (*Sub-rule* (*shuffle*))**.** Let $S$ be a decomposed ZCUP, and let $L$ be a $\psi$-minimal path-unique Z-cycle of length $h \geqslant 2$. Select an index $j$ such that $C_j$ is not trivial.

Select one of the following possibilities.

(1) Select some $i$ and replace $X_i$ by $Id$ or by $K\ x$, where $x$ is new. Then standardize the resulting ZCUP.

(2) Let $C_j[X_{j+1 \bmod* h}(t_j)]$ have the form

$$f(t_{j,1}, \ldots, t_{j,k-1}, t_{j,k}\lfloor X_{j+1 \bmod* h}\rfloor, t_{j,k+1}, \ldots, t_{j,m}).$$

(a) Select an index $1 \leqslant r \leqslant m$.

(b) Instantiate $X_j$ by $f(x_1, \ldots, x_{r-1}, X'_j(\cdot), x_{r+1}, \ldots, x_m)$, where the $x_i, X'_j$ are new.

(c) Apply (decomp) to the $j$th equation of $L$ after the instantiation, i.e., to

$$f(x_1, \ldots, x_{r-1}, X'_j(s_j), x_{r+1}, \ldots, x_m)$$
$$\doteq$$
$$f(t_{j,1}, \ldots, t_{j,k-1}, t_{j,k}\lfloor X_{j+1 \bmod* h}\rfloor, t_{j,k+1}, \ldots, t_{j,m}).$$

(d) Apply (repvt) or (repvv) for all equations $x_i \doteq t_{j,i}$ for $i \neq r$ added by the last step.

(e) Then standardize the resulting ZCUP.

The rule (shuffle) basically has two possibilities:

If $r \neq k$, then a shorter Z-cycle will be generated.

If $r = k$, then it is a very shuffle step: syntactically there is no real progress, since symbols are shuffled from one place to the other.

**Lemma 6.7.** *The rule* (*shuffle*) *is sound and complete for ZCUPs of type* 3.

**Lemma 6.8.** *Let $S$ be a decomposed ZCUP and $L$ be a path-unique, $\psi$-minimal Z-cycle of length $h > 1$. Let $S'$ be obtained from $S$ by a non-failing application of the rule* (*shuffle*). *Either we have* $(\mu_1(S'), \mu_{2,1}(S'), \mu_{2,2}(S')) < (\mu_1(S), \mu_{2,1}(S), \mu_{2,2}(S))$, *where the comparison is lexicographically, or* (*in the case $r = k$*) *$S'$ is decomposed, contains the same number of second order variables, and contains a path-unique Z-cycle $L'$ of length $h$ such that the relevant contexts $C'_1, \ldots, C'_h$ have main depths corresponding to the contexts $C_1, \ldots, C_h$ of $L$ except for positions $j$ and $j-1$ mod∗ $h$. We have* $|C'_{j-1 \bmod* h}| = |C_{j-1 \bmod* h}| + 1$ *and* $|C'_j| = |C_j| - 1$.

**Proof.** See Appendix A.2. □

**Definition 6.9** (*shuffle**)**.** Let $S$ be a ZCUP and let $L$ be a $\psi$-minimal path-unique Z-cycle of length $h \geqslant 2$ with at least two nontrivial relevant contexts $C_j$ and $C_{j'}$, where $j \neq j'$. Let $\bar{\mu}$ be $\mu(S)$ before (shuffle*) starts.

Then iterate (shuffle) as follows:

(1) First select an index $j$ in the Z-cycle, such that $C_j$ is nontrivial.

(2) Apply (shuffle) for index $j$.
(3) If $\mu(S')$ is strictly smaller than $\bar{\mu}$, then stop and return $S'$.
(4) Otherwise, let $L'$ be the Z-cycle obtained from $L$. If $C'_j$ is nontrivial, then go to 2 operating on $L'$ using the same index.
If $C'_j$ is trivial, then go to 2 using the index $j-1\ mod* h$ instead of $j$, and the Z-cycle $L'$.

Note that $\mu_{2,3}$ may be temporarily increased to $\bar{\mu}_{2,3}+1$ by a step within (shuffle*), if the relevant context $C_{j-1\ mod* h}$ is trivial. An intermediate state is that $C_j$ as well as $C_{j-1\ mod* h}$ are nontrivial. After $|C_j|$ steps, the relevant context at index $j$ is trivial, whereas the relevant context at index $j-1\ mod* h$ is nontrivial. The goal of (shuffle*) is to move the relevant contexts until two of them are merged. In this case the number of relevant contexts in the length-minimal Z-cycle is decreased.

**Lemma 6.10.** *The rule (shuffle*) is sound and complete for ZCUPs of type* 3 *with* $\mu_{2,3} \geqslant 2$.

*Let $S$ be a ZCUP of type* 3 *with* $\mu_{2,3} \geqslant 2$, *i.e., that contains a $\psi$-minimal, path-unique Z-cycle $L$ with at least two nontrivial relevant contexts $C_j$, $C_{j'}$ with $j \neq j'$. Then (shuffle*) either fails or results in a ZCUP $S'$ of types* 0–3 *such that* $\mu(S') < \mu(S)$.

**Proof.** Soundness is standard. Completeness follows from completeness of (shuffle), see Lemma 6.7, and since the algorithm (shuffle*) applies (shuffle) only if the preconditions are met.

We may reach, by an iterated application of the rule (shuffle), a ZCUP $S'$ with $(\mu_1(S'), \mu_{2,1}(S'), \mu_{2,2}(S')) < (\bar{\mu}_1, \bar{\mu}_{2,1}, \bar{\mu}_{2,2})$ (lexicographically), see Lemma 6.8. In the other case, using Lemma 6.8, it is clear that after at most $(h-1)*|C_j|$ iterations, two of the relevant contexts are merged into one relevant context. Hence in this case after several iterations, $(\mu_1(S'), \mu_{2,1}(S'), \mu_{2,2}(S')) = (\bar{\mu}_1, \bar{\mu}_{2,1}, \bar{\mu}_{2,2})$, but $\mu_{2,3}(S') < \bar{\mu}_{2,3}$, and thus $\mu(S') < \bar{\mu}$. □

Finally we describe the reduction of ZCUPs $L$ of type 3 in the situation where some $\psi$-minimal, compressed and path-unique Z-cycle $L$ contains just one nontrivial relevant context $C_j$. We may assume that $j = h$ and $L$ has the form $X_1(s_1) \doteq X_2(t_1), \ldots, X_{h-1}(s_{h-1}) \doteq X_h(t_{h-1}), X_h(s_h) \doteq C_h[X_1(t_h)]$, where $C_h$ is nontrivial.

In order to avoid instantiating a second order variable $X_i$ by a context that contains $X_i$, the following construction is defined:

Let $C$ be a context. The *skeleton context* $B$ of $C$ is constructed as follows: If $C \equiv f(t_1, \ldots, t_{i-1}, C', t_{i+1}, \ldots, t_m)$, then $B \equiv f(x_1, \ldots, x_{i-1}, B', x_{i+1}, \ldots, x_m)$, where $B'$ is the skeleton context of $C'$ and the variables $x_i$ are fresh ones. For the empty context, the skeleton is the empty context. For example, the skeleton context of $f(g(a,b), g(b,[\cdot],c))$ is $f(x, g(y,[\cdot],z))$.

Recall the following:
- $E$ is the upper bound on the exponent of periodicity fixed in the algorithm $UZ$.
- $C^e$ for a context $C$ and a positive integer $e$ means the expanded form $\underbrace{C \ldots C}_{e}$.
- *firstdpos*$(C)$ is the notation of the first digit of the position of the hole of a nontrivial context $C$.

**Definition 6.11** (*solve-unique-Z-cycle*)**.** Input is a ZCUP $S$ of type 3, such that there is a $\psi$-minimal, compressed and path-unique Z-cycle $L$ that has exactly one nontrivial relevant context. Select such

a $\psi$-minimal path-unique Z-cycle $L$ of length $h$ with nontrivial relevant context $C_h$. Select one of the following possibilities.

(1) Select some second order variable $X_i$ and replace $X_i$ by $Id$ or by $K\ x$, where $x$ is new. Then standardize the resulting ZCUP.

(2) Construct the skeleton context $B_h$ of $C_h$. Select some $0 \leqslant e \leqslant E$ and some (possibly trivial) prefix $B_{h,1}$ of $B_h$, i.e. $B_h \equiv B_{h,1}B_{h,2}$ for some $B_{h,2}$. Replace each $X_i$ either by $B_h^e B_{h,1}[X_i'(\cdot)]$, or by $B_h^e B_{h,1}$, where $X_i'$ is new. For at least one index $i$, the second case should be selected. Then standardize the resulting ZCUP.

(3) This selection is only applicable if $h > 1$. Construct the skeleton context $B_h$ of $C_h$.
   (a) Select $0 \leqslant e \leqslant E$ and some (possibly trivial) proper prefix $B_{h,1}$ of $B_h$, such that $B_h \equiv B_{h,1}B_{h,2}$ and $B_{h,2}B_{h,1}$ has a top level function symbol $f$ of arity $n > 1$. If this is not possible, then fail.
   (b) Select for every $j$ with $1 \leqslant j \leqslant h$ an index $k_j$ with $1 \leqslant k_j \leqslant n$ and replace $X_j$ by $B_h^e B_{h,1} f(x_{j,1}, \ldots, x_{j,k_j-1}, X_j'(\cdot), x_{j,k_j+1}, \ldots, x_{j,n})$ with new variables. Let $k := \mathit{firstdpos}\ (B_{h,2}B_{h,1})$. At least one index $k_j$ should be different from $k$.
   (c) Apply (decomp) to the equations obtained from the equations of $L$ by instantiation.
   (d) From the last step, among others, the following $h$ equations are obtained.

$$\begin{aligned} f(x_{1,1}, \ldots, X_1'(s_1'), \ldots, x_{1,n}) &\doteq f(x_{2,1}, \ldots, X_2'(t_1'), \ldots, x_{2,n}) \\ \ldots &\doteq \ldots \\ f(x_{h-1,1}, \ldots, X_{h-1}'(s_{h-1}'), \ldots, x_{h-1,n}) &\doteq f(x_{h,1}, \ldots, X_h'(t_{h-1}'), \ldots, x_{h,n}) \\ f(x_{h,1}, \ldots, X_h'(s_h'), \ldots, x_{h,n}) &\doteq B_{h,2}B_{h,1} f(x_{1,1}, \ldots, X_1'(t_h'), \ldots, x_{1,n}) \end{aligned}$$

   Apply (decomp) once to every equation, and after this apply (repvt) or (repvv) to the equations obtained from (decomp) at index $k$ of every equation.
   (e) Then standardize the resulting ZCUP.

**Lemma 6.12.** *The rule (solve-unique-Z-cycle) is sound and complete for ZCUPs of type* 3 *with* $\mu_{2,3} = 1$.

*If $S$ be a problem of type* 3 *with* $\mu_{2,3} = 1$, *i.e., having a $\psi$-minimal, compressed and path-unique Z-cycle with exactly one nontrivial relevant context, then application of* (*solve-unique-Z-cycle*) *either fails, or the output is a ZCUP $S'$ with* $\mu(S') < \mu(S)$.

**Proof.** see Appendix A.3 □

**Example 6.13.** We demonstrate the use of (solve-unique-Z-cycle) in the algorithm *UZ* for the ZCUP of type 3:

$$\{X(Y(a)) \doteq f(Z(X(b)), X(c))\}.$$

One possibility is that case 2. is selected with $e = 2$ and $X$ is instantiated by $f(x, f(x, [\cdot]))$:

$$\{f(x, f(x, Y(a))) \doteq f(Z(f(x, f(x, b))), f(x, f(x, c)))\}.$$

Two applications of (decomp) and an application of (repvv) yield:

$$\{x \doteq Z(f(x, f(x, b))),\ Y(a) \doteq f(x, c)\}.$$

The ZCUP has to be standardized:

$$\{x \doteq Z(f(x, f(x, b))),\ Y(a) \doteq f(x, y_1), y_1 \doteq c\}.$$

The ZCUP is of type 1. Using (imitation), we select the maximal equivalence class $\{Y\}$. There are two possibilities for the second equation. We choose $Y := f(Y'(\cdot), y_2)$. After (decomp) and (repvv), this results in:

$$\{x \doteq Z(f(x, f(x, b))),\ Y'(a) \doteq x, y_2 \doteq c\}.$$

An imitation step and subsequent application of (decomp) has the effect to remove the equation $y_2 \doteq c$. The result after standardization is

$$\{x \doteq Z(f(x, f(x, b))),\ x \doteq Y'(a)\}.$$

Now the ZCUP is of type 0, and hence it is unifiable.

**Example 6.14.** Consider the following ZCUP with "connected" Z-cycles:

$$\begin{array}{l}\{X_1(t_1) \doteq X_2(s_1)\\ X_2(t_2) \doteq f(X_1(s_2), X_3(s_3))\\ X_3(t_3) \doteq X_2(s_4)\}.\end{array}$$

Then there are two length-minimal Z-cycles. The ZCUP is of type 3. The rule (solve-unique-Z-cycle) will select one of them, say the $X_1, X_2$-cycle. We demonstrate some of the nontrivial possibilities:

(1) Instantiation not following the positions of the "cycling variables", i.e. application of case 3. using the trivial prefix context of the skeleton context $f([\cdot], y)$ using the instantiation

$$X_1 \to f(x_1, X_1'(\cdot)), \quad X_2 \to f(x_2, X_2'(\cdot))$$

results in:

$$\begin{array}{ll}\{x_1 & \doteq x_2\\ X_1'(t_1) & \doteq X_2'(s_1)\\ x_2 & \doteq f(x_1, X_1'(s_2))\\ X_2'(t_2) & \doteq X_3(s_3)\\ \multicolumn{2}{c}{\ldots} & \}.\end{array}$$

Application of (repvt) according to (3d) results in an occurs-check failure during standardization.

(2) Another instantiation not following the positions of the "cycling variables" using case 3:

$$X_1 \to f(X_1'(\cdot), x_1), \quad X_2 \to f(x_2, X_2'(\cdot)).$$

Result:

$$\begin{array}{ll}\{X_1'(t_1) & \doteq x_2\\ x_1 & \doteq X_2'(s_1)\\ x_2 & \doteq f(X_1'(s_2), x_1)\\ X_2'(t_2) & \doteq X_3(s_3)\\ & \ldots \qquad\qquad \}.\end{array}$$

Application of (repvt) according to (3d) gives a shorter compressed Z-cycle:

$$X_1'(t_1) \doteq f(X_1'(s_2), x_1).$$

(3) Instantiations following the cycle would usually result in ZCUPs that are not smaller w.r.t. $\mu$:

$$X_1 \to f(X_1'(\cdot), x_1), \quad X_2 \to f(X_2'(\cdot), x_2)$$

Result:

$$\begin{array}{ll}\{X_1'(t_1) & \doteq X_2'(s_1)\\ x_1 & \doteq x_2\\ X_2'(t_2) & \doteq f(X_1'(s_2), x_1)\\ x_2 & \doteq X_3(s_3)\\ & \ldots \qquad\qquad \}.\end{array}$$

However, the rule (solve-unique-Z-cycle) permits in this case only instantiations that either strictly decrease the number of Z-context variables, or instantiations that simulate a limited number of instantiations round the Z-cycle and then a deviation, which results in failure or a shorter Z-cycle.

## 7. Bounded second order unification is $\mathcal{NP}$-hard

An instance of a (positive) ONE-IN-THREE-SAT problem is a set of propositional clauses, without negation symbols, where every clause has three propositional variables. The question is whether there is an assignment of truth values to the propositional variables, such that in every clause there is exactly one propositional variable that is made true by the assignment.

It is well-known that positive ONE-IN-THREE-SAT is $\mathcal{NP}$-complete [9]. We construct an encoding of every instance of ONE-IN-THREE-SAT as a BSOUP:

Given an instance of the positive ONE-IN-THREE-SAT problem $p_{\varphi(i,1)} \vee p_{\varphi(i,2)} \vee p_{\varphi(i,3)}$, $i = 1, \ldots, n$, where $p_j$ for $j = 1, \ldots, m$ are the propositional variables, we construct the following bounded second order unification problem.

Let $X_k$ for $k = 1, \ldots, m$ be second order variables, let $g$ be a unary function symbol and $a$ be a constant. The equations are

$$\begin{array}{c}\{X_j(g(a)) \doteq g(X_j(a)) \mid j = 1, \ldots, m\}\\ \cup\\ \{X_{\varphi(i,1)}(X_{\varphi(i,2)}(X_{\varphi(i,3)}(a))) \doteq g(a) \mid i = 1, \ldots, n\}.\end{array}$$

The translation of truth is as follows: $p_j$ is true if the unifier instantiates $X_j$ by $g([\cdot])$, and false otherwise.

Now we show that solvability of the instance of ONE-IN-THREE-SAT is equivalent to unifiability of the BSOU. It is bounded, since the signature is monadic.

The first type of equations implies that every unifier must have components of the form $X_j \to g^{m_j}([\cdot])$ for some nonnegative integer $m_j$ for $j = 1, \ldots, m$.

The second type of equations implies that $m_j \in \{0, 1\}$ for all $j$. Moreover, the second type of equations implies that for every $i$ exactly one of the variables $X_{\varphi(i,1)}, X_{\varphi(i,2)}, X_{\varphi(i,3)}$ is instantiated by $g([\cdot])$. Now it is easy to verify that the ONE-IN-THREE-SAT problem is solvable iff the constructed Z-context unification problem has a unifier.

It is also easy to verify that the encoding is polynomial.

**Proposition 7.1.** *Z-context unification is $\mathcal{NP}$-hard.*

**Theorem 7.2.** *Bounded second order unification is $\mathcal{NP}$-hard.*

For monadic second order unification (MSOU) which is second order unification where the signature has only monadic function symbols the same encoding can be used, hence the following is immediate.

**Corollary 7.3.** *Monadic second order unification is $\mathcal{NP}$-hard.*

## 8. Conclusion

This paper shows that restricting the number of holes in instantiations of second order variables makes second order unification decidable. It proves that the undecidability encoding of second order unification requires an unbounded number of variable occurrences in the unifier. From a more practical viewpoint, it permits a semi-decision procedure for unifiability of second order unification problems by successively increasing the bound and calling the decision procedure for every bound.

This result is consistent with the (yet unproven) hypothesis that context unification is decidable. However, the methods and the results seem not helpful in solving the context unification problem. The algorithm could be adapted to transform context unification problems into a type-0 form, but there is no decision method known for context unification problems of type 0.

Unfortunately, the decision algorithm for ZCUPs is rather complex, it requires at least exponential space, since the rule (solve-unique-Z-cycle) may generate an exponential number of copies of a context. Nevertheless, there is a potential for optimizations, in particular, the author conjectures that the decision problem for BSOU is $\mathcal{NP}$-complete.

## Acknowledgments

I thank Klaus Schulz for his numerous comments, and also Jordy Levy and Mattieu Villaret. The comments of the referees, in particular of the anonymous referee B, helped to eliminate several errors in previous versions and also helped to improve the presentation.

## Appendix

This appendix contains proofs of completeness and termination in terms of $\mu$ of the main reduction rules. Proofs of soundness are omitted, since they are straightforward.

### *A.1. Proof of Lemma 6.4*

then $\mu(S') < \mu(S)$.

Clearly the output problem is of types 0–3.

We show completeness for ZCUPs of type 2. Let $\sigma$ be a unifier of $S$. If $\sigma(X_i)$ is $Id$ or $K\ s$ for some $s$ and for some $1 \leqslant i \leqslant h$, then use selection 1. Let us now assume that none of the expressions $\sigma(X_i)$ is the identity or $K\ s$ for $1 \leqslant i \leqslant h$.

*Case* $h > 1$: Let $j$ be the selected index and $r = \mathit{firstdpos}\ \ (\sigma(X_j))$. Obviously $r$ represents a possible choice in selection 2, and it is straightforward to check that a ZCUP $S'$ that is reached after instantiation and decomposition has a unifier $\sigma'$ with exponent of periodicity $\leqslant E$.

*Case* $h = 1$: Note that $\sigma(X_1)$ is assumed to be nontrivial. Here $L$ has the form $X_1(s_1) \doteq f(t_{1,1}, \ldots, t_{1,m})\lfloor X_1 \rfloor$. We claim that unifiability implies that all surface occurrences of $X_1$ in $f(t_{1,1}, \ldots, t_{1,m})$ are in the unique subterm $t_{1,r}$ where the index $r$ is $\mathit{firstdpos}(\sigma(X_1))$. Assume that there exists an index $1 \leqslant k \leqslant m, k \neq r$ such that $t_{1,k}$ has a surface occurrence of $X_1$. The definition of $r$ and the form of $L$ show that $\sigma(t_{1,k})$ is a proper subterm of $\sigma(X_1)$. On the other hand, $\sigma(t_{1,k})$ contains an occurrence of $\sigma(X_1)$. This is a contradiction. We have seen that the index $r$ represents a (the only) possible selection. Obviously the ZCUP $S'$ that is reached after instantiation and decomposition has a unifier $\sigma'$ with exponent of periodicity $\leqslant E$.

Let us now verify $\mu(S') < \mu(S)$. This is obvious if the first selection is used since the number of second order variables is strictly decreased.

Now assume that the second selection has been chosen. Note that length-minimality implies that the Z-cycle contains only the explicitly indicated surface occurrences of $X_i$ for $i = 1, \ldots, h$ (see Lemma 4.11). In this proof we omit in the notation the more rigorous $mod* h$, and assume that the indices are $\in \{1, \ldots, h\}$.

First consider the case where $h \geqslant 3$. Then $L$ contains a subsequence

$$\begin{aligned} X_{j-1}(s_{j-1}) &\doteq t_{j-1}\lfloor X_j \rfloor, \\ X_j(s_j) &\doteq f(t_{j,1}, \ldots, t_{j,m})\lfloor X_{j+1} \rfloor, \\ X_{j+1}(s_{j+1}) &\doteq t_{j+1}, \end{aligned}$$

where $X_{j+1}$ has at least two surface occurrences in $t_j \equiv f(t_{j,1}, \ldots, t_{j,m})$.

Instantiation (2b) leads to the equations

$$\begin{aligned} X_{j-1}(s'_{j-1}) &\doteq t'_{j-1}\lfloor f(x_1, \ldots, x_{r-1}, X'_j(\cdot), x_{r+1}, \ldots, x_m) \rfloor, \\ f(x_1, \ldots, x_{r-1}, X'_j(s'_j), x_{r+1}, \ldots, x_m) &\doteq f(t'_{j,1}, \ldots, t'_{j,m})\lfloor X_{j+1} \rfloor, \\ X_{j+1}(s'_{j+1}) &\doteq t'_{j+1}, \end{aligned}$$

(where terms $t'$ are obtained from $t$ by instantiation). Application of the rule (decomp) in step (2c) yields

$$X_{j-1}(s'_{j-1}) \doteq t'_{j-1}\lfloor f(x_1, \ldots, x_{r-1}, X'_j(\cdot), x_{r+1}, \ldots, x_m) \rfloor,$$

$$X'_j(s'_j) \doteq t'_{j,r},$$
$$X_{j+1}(s'_{j+1}) \doteq t'_{j+1}$$

plus the equations $x_i \doteq t'_{j,i}$ for $i \neq r$. Application of (repvt) or (repvv) in Step (2d) leads to the ZCUP $S'$ containing the following three equations

$$X_{j-1}(s'_{j-1}) \doteq t'_{j-1}\lfloor f(t'_{j,1}, \ldots, t'_{j,r-1}, X'_j(\cdot), t'_{j,r+1}, \ldots, t'_{j,m})\rfloor, \tag{A.1}$$
$$X'_j(s'_j) \doteq t'_{j,r}, \tag{A.2}$$
$$X_{j+1}(s'_{j+1}) \doteq t'_{j+1}. \tag{A.3}$$

First assume that there is at least one index $k \neq r$ such that $t_{j,k}$ has a surface occurrence of $X_{j+1}$. We have that $X_{j+1} \not\equiv X_j$, since $h > 1$, and the Z-cycle has minimal length. Hence $t'_{j,k}$ has a surface occurrence of $X_{j+1}$. This shows that the equations

$$X_{j-1}(s'_{j-1}) \doteq t'_{j-1}\lfloor f(t'_{j,1}, \ldots, t'_{j,k}\lfloor X_{j+1}\rfloor, \ldots, t'_{j,r-1}, X'_j(\cdot), t'_{j,r+1}, \ldots, t'_{j,m})\rfloor,$$
$$X_{j+1}(s'_{j+1}) \doteq t'_{j+1}$$

together with the images of the equations of $L$ with indices $\notin \{j, j+1, j-1\}$ represent a Z-cycle $L'$ of length $h-1$. Note that the conditions for a Z-cycle are satisfied, since $t'_{j-1}\lfloor f(t'_{j,1}, \ldots, t'_{j,k}\lfloor X_{j+1}\rfloor, \ldots, t'_{j,r-1}, X'_j(\cdot), t'_{j,r+1}, \ldots, t'_{j,m})\rfloor$ contains a function symbol as head. Standardization does not increase the $\psi$-measure of the Z-cycle by Lemma 4.28 and the resulting ZCUP $S''$ contains a Z-cycle $L''$ such that $\psi(L'') = \psi(L') < \psi(L)$, which shows that $\mu(S'') < \mu(S)$.

In the other case where $r = k$, the term $t_{j,r}$ – and hence $t'_{j,r}$ – contains at least two surface occurrences of $X_{j+1}$. The three equations (A.1)–(A.3) above can be combined with the images of the equations of $L$ with indices $\notin \{j, j+1, j-1\}$ to a new Z-cycle $L'$. When moving from $L$ to $L'$, the main depth of the relevant context of equation $j$ is decreased. Since $t'_{j,r}$ contains at least two surface occurrences of $X_{j+1}$, the new Z-cycle $L'$ is non-path-unique and this main depth is relevant for the $\psi$-measure.

It follows that $\psi(L') < \psi(L)$. As in the previous case it follows now that a ZCUP $S''$ is reached such that $\mu(S'') < \mu(S)$.

The arguments for case $h = 2$ can be obtained by specializing the arguments for $h > 2$. The first and the third equation in the considered subsequence of $L$ in the case $h > 2$ are identical in this case. However, all other arguments are the same.

Consider the special situation where $h = 1$. Here $L$ has the form $X_1(s_1) \doteq f(t_{1,1}, \ldots, t_{1,m})\lfloor X_1\rfloor$. There are at least two surface occurrences of $X_1$ in $t_1 \equiv f(t_{1,1}, \ldots, t_{1,m})$, and all these surface occurrences are in $t_{1,r}$. Instantiation (2b) yields $f(x_1, \ldots, x_{r-1}, X'_1(s'_1), x_{r+1}, \ldots, x_m) \doteq f(t'_{1,1}, \ldots, t'_{1,m})\lfloor f(x_1, \ldots, x_{r-1}, X'_1, x_{r+1}, \ldots, x_m)\rfloor$. step (2c) yields the new Z-cycle $L'$ with the equation $X'_1(s'_1) \doteq t'_{1,r}$ (see Example 6.5 part 3). The main depth of the relevant context of $L'$ is smaller than the main depth of the relevant context for $L$. Hence $\psi(L') < \psi(L)$. As in the previous cases we see that after Step (2d) a problem $S''$ is reached such that $\mu(S'') < \mu(S)$. □

*A.2. Proof of Lemma 6.8*

If we use selection 1, then the number of second order variables is decreased. Assume now that we have chosen selection 2.

Again we omit the $mod* h$ to adjust indices correctly to simplify notation. As in the previous proof, only the case $h = 2$ requires special attention to check that this is no problem.

First consider the case where $r \neq k$. Let the equations with indices $j-1, j, j+1$ of $L$ have the form

$$\begin{aligned} X_{j-1}(s_{j-1}) &\doteq C_{j-1}[X_j(t_{j-1})], \\ X_j(s_j) &\doteq f(t_{j,1}, \ldots, t_{j,k-1}, t_{j,k}\lfloor X_{j+1}\rfloor, t_{j,k+1}, \ldots, t_{j,m}), \\ X_{j+1}(s_{j+1}) &\doteq t. \end{aligned}$$

Note that path-uniqueness and length-minimality imply that the Z-cycle contains only the explicitly indicated surface occurrences of $X_i$ (see Lemma 4.11). Instantiation (2b) leads to

$$\begin{aligned} X_{j-1}(s'_{j-1}) &\doteq C'_{j-1}[f(x_1, \ldots, x_{r-1}, X'_j(t'_{j-1}), x_{r+1}, \ldots, x_m)], \\ f(x_1, \ldots, x_{r-1}, X'_j(s'_j), x_{r+1}, \ldots, x_m) &\doteq f(t'_{j,1}, \ldots, t'_{j,k-1}, t'_{j,k}\lfloor X_{j+1}\rfloor, t'_{j,k+1}, \ldots, t'_{j,m}), \\ X_{j+1}(s'_{j+1}) &\doteq t' \end{aligned}$$

Decomposition (2c) and replacement steps (2d) lead to a ZCUP with a Z-cycle $L'$ of length $h-1$ with the equations

$$\begin{aligned} X_{j-1}(s'_{j-1}) &\doteq C'_{j-1}[f(t'_{j,1}, \ldots, X'_j(t'_{j-1}), \ldots, t'_{j,k}\lfloor X_{j+1}\rfloor, \ldots, t'_{j,m})], \\ X_{j+1}(s'_{j+1}) &\doteq t'. \end{aligned}$$

This means that $\mu_{2,1}(S') < \mu_{2,1}(S)$, which also holds after standardization. Hence we have $(\mu_1(S'), \mu_{2,1}(S'), \mu_{2,2}(S')) < (\mu_1(S), \mu_{2,1}(S), \mu_{2,2}(S))$.

Now consider the selection $r = k$. In this case, instantiation (2b) leads to

$$\begin{aligned} X_{j-1}(s'_{j-1}) &\doteq C'_{j-1}[f(x_1, \ldots, x_{k-1}, X'_j(t'_{j-1}), x_{k+1}, \ldots, x_m)], \\ f(x_1, \ldots, x_{k-1}, X'_j(s'_j), x_{k+1}, \ldots, x_m) &\doteq f(t'_{j,1}, \ldots, t'_{j,k-1}, t'_{j,k}\lfloor X_{j+1}\rfloor, t'_{j,k+1}, \ldots, t'_{j,m}), \\ X_{j+1}(s'_{j+1}) &\doteq t' \end{aligned}$$

Decomposition (2c) and replacement steps (2d) lead to a ZCUP with a variant $L'$ of the Z-cycle $L$, of length $h$, where the equations with indices $j-1, j, j+1$ have the form

$$\begin{aligned} X_{j-1}(s'_{j-1}) &\doteq C'_{j-1}[f(\ldots, X'_j(t'_{j-1}), \ldots)], \\ X'_j(s'_j) &\doteq t'_{j,k}\lfloor X_{j+1}\rfloor, \\ X_{j+1}(s'_{j+1}) &\doteq t'. \end{aligned}$$

The new Z-cycle $L'$ is path-unique, and it is easy to see that it satisfies the properties mentioned in the lemma. □

*A.3. Proof of Lemma 6.12*

We show completeness for ZCUPs of type 3 with $\mu_{2,3} = 1$: Let $\sigma$ be a unifier of $S$ with an exponent of periodicity that is not greater than $E$. If $\sigma$ instantiates some second order variable in the Z-cycle $L$ as $Id$ or by $K\ s$, then use selection 1.

Otherwise, let $C_0$ be the greatest common prefix of the contexts $\sigma(X_j), j = 1, \ldots, h$, and of $\sigma(C_h)^E$. If there is some $X_j$, such that $\sigma(X_j) = C_0$, then case 2. can be selected, since $C_0$ is a prefix of $\sigma(C_h)^E$. Note that for an extension $\sigma'$ (on new variables) of $\sigma$, $\sigma'(B_h) = \sigma(C_h)$.

The remaining case is that $C_0$ is a proper prefix of all $\sigma(X_j)$, where $j = 1, \ldots, h$.

First we show that this is not possible for $h = 1$: Assume otherwise. Let $C_1 \equiv C_{11}C_{12}$, such that $\sigma(C_1^e C_{11}) \equiv C_0$. For some index $k \neq \mathit{firstdpos}\ (C_{12}C_{11})$, we have $\sigma(X_1) = \sigma(C_1)^e \sigma(C_{11}) f(r_1, \ldots, r_k \lfloor \cdot \rfloor, \ldots, r_n)$. The equation $X_1(s_1) \doteq C_1[X_1(t_1)]$ after applying $\sigma$ is

$$\begin{array}{c} \sigma(C_1)^e \sigma(C_{11}) f(r_1, \ldots, r_k[\sigma(s_1)], \ldots, r_n) \\ \doteq \\ \sigma(C_1)\sigma(C_1)^e \sigma(C_{11}) f(r_1, \ldots, r_k \lfloor \sigma(s_1) \rfloor, \ldots, r_n). \end{array}$$

This implies that the following equation holds:

$$f(r_1, \ldots, r_k \lfloor \sigma(s_1) \rfloor, \ldots, r_n) \equiv \sigma(C_{12}C_{11}) f(r_1, \ldots, r_k \lfloor \sigma(s_1) \rfloor, \ldots, r_n).$$

Let $\sigma(C_{12}C_{11}) = f(a_1, \ldots, a_{j-1}, \underbrace{D[\cdot]}_{j}, a_{j+1}, \ldots, a_n)$, where $\mathit{firstdpos} = j \neq k$. Then by decomposition, we get that $r_j \equiv D[f(r_1, \ldots, r_k \lfloor \sigma(s_1) \rfloor, \ldots, r_n)]$, which is impossible.

Now we can assume that $h \geqslant 2$. There is a non-unary function symbol $f$, such that $\sigma(X_j) = C_0 D_j$, and $f$ is the top level function symbol of $D_j$ for $j = 1, \ldots, h$. Case 3. can be selected. We select $B_{h,1}, B_{h,2}$ such that $C_0 = \sigma(B_h^e B_{h,1})$, where $e \leqslant E$. The function symbol $f$ is then also the head symbol of $B_{h,2}B_{h,1}$. The arity of $f$ is greater than 1, since $C_0$ is a greatest common prefix. In (3b) we choose $k_j$ to be $\mathit{firstdpos}(D_j)$. It follows from maximality of the common prefix $C_0$, that at least one $k_j$ is different from $\mathit{firstdpos}\ (B_{h,2}B_{h,1})$. Constructing the new unifier $\sigma'$ is done as follows: Let $\sigma'$ be such that $\sigma'(B_h) = \sigma(C_h)$. The further construction of $\sigma'$ is straightforward. The exponent of periodicity of $\sigma'$ is not greater than $E$.

The measure $\mu$ is strictly decreased:

This is obvious in the first two cases, since the number of second order variables is strictly decreased. If selection 3 is selected we show that if no fail occurs, then a new Z-cycle is generated whose length is strictly shorter than $h$. Note that we can assume that $h \geqslant 2$. We make the proof for the case $h \geqslant 3$, the case for $h = 2$ can be obtained by specialization. Note that the contexts $B_h, B_{h,1}, B_{h,2}$ do not contain any occurrences of the second order variables $X_i$.

$L$ has the form

$$X_1(s_1) \doteq X_2(t_1), \ldots, X_h(s_h) \doteq C_h[X_1(t_h)].$$

Instantiation (3b) yields the equations

$$\begin{array}{c} B_h^e B_{h,1} f(x_{1,1}, \ldots, \underbrace{X_1'(s_1')}_{k_1}, \ldots, x_{1,n}) \doteq B_h^e B_{h,1} f(x_{2,1}, \ldots, \underbrace{X_2'(t_1')}_{k_2}, \ldots, x_{2,n}), \\ \ldots \doteq \ldots \\ B_h^e B_{h,1} f(x_{h,1}, \ldots, \underbrace{X_h'(s_h')}_{k_h}, \ldots, x_{h,n}) \doteq C_h' B_h^e B_{h,1} f(x_{1,1}, \ldots, \underbrace{X_1'(t_h')}_{k_1}, \ldots, x_{1,n}). \end{array}$$

Several applications of (decomp) yield (among others) the equations:

$$f(x_{1,1},\ldots,\underbrace{X'_1(s'_1)}_{k_1},\ldots,x_{1,n}) \doteq f(x_{2,1},\ldots,\underbrace{X'_2(t'_1)}_{k_2},\ldots,x_{2,n}),$$
$$\ldots \doteq \ldots$$
$$f(x_{h-1,1},\ldots,\underbrace{X'_{h-1}(s'_{h-1})}_{k_{h-1}},\ldots,x_{h-1,n}) \doteq f(x_{h,1},\ldots,\underbrace{X'_h(t'_{h-1})}_{k_h},\ldots,x_{h,n}),$$
$$f(x_{h,1},\ldots,\underbrace{X'_h(s'_h)}_{k_h},\ldots,x_{h,n}) \doteq B_{h,2}B_{h,1}f(x_{1,1},\ldots,\underbrace{X'_1(t'_h)}_{k_1},\ldots,x_{1,n}).$$

Let $k$ be $\mathit{firstdpos}(B_{h,2}B_{h,1})$. Then consider all the equations that result from applying (decomp) to the equations, where only the result at index $k$ is considered. Let $B_{h,3}$ be such that $B_{h,2}B_{h,1} \equiv B_{h,4}B_{h,3}$ for some context $B_{h,4}$ of main depth 1. The following pairs of equations are the result: every equation for $2 \leqslant j < h$ is of the form

$$s''_{j-1,k} \doteq t''_{j-1,k}$$
$$s''_{j,k} \doteq t''_{j,k},$$

where either $t''_{j-1,k} \equiv s''_{j,k} \equiv x_{j,k}$, or $t''_{j-1,k} \equiv X'_j(t'_{j-1}), s''_{j,k} \equiv X'_j(s'_j)$.

The equation for the index $h$ has two possibilities: either

$$x_{h,k} \doteq B_{h,3}[f(x_{1,1},\ldots,\underbrace{X'_1(t'_h)}_{k_1},\ldots,x_{1,n})]$$

or

$$X'_h(s'_h) \doteq B_{h,3}[f(x_{1,1},\ldots,\underbrace{X'_1(t'_h)}_{k_1},\ldots,x_{1,n})].$$

There are two possibilities: Either there are only equations between variables for all $j < h$. In this case there is a fail due to occurs-check for the equations at index $k$, which will be detected in the decomposition step of standardization. This happens, if $k_j \neq k$ for all $j$.

The other case is that there is at least one equation for $j < h$ where $s_{j,k} \equiv X'_j(s'_j)$. Then at index $k$, the chain of equations is a Z-cycle of length $h$. Moreover, there is at least one first order variable in the chain, since there is some $j$, such that $k_j \neq k$. Using (repvt) for the corresponding indices $j < h$ results in a compressed Z-cycle of length $\leqslant h-1$. The subsequent standardization does not increase the length of the shortest cycle.

Hence in the non-failing cases, we have $\mu(S') < \mu(S)$. □

## References

[1] F. Baader, J. Siekmann, Unification theory, in: D. Gabbay, C. Hogger, J. Robinson (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Oxford University Press, Oxford, 1994, pp. 41–125.

[2] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, Cambridge, 1998.

[3] H. Comon, Completion of rewrite systems with membership constraints. Part I: Deduction rules, J. Symbolic Comput. 25 (4) (1998) 397–419.
[4] H. Comon, Completion of rewrite systems with membership constraints. Part II: Constraint solving, J. Symbolic Comput. 25 (4) (1998) 421–453.
[5] N. Dershowitz, Z. Manna, Proving termination with multiset orderings, Commun. ACM 22 (1979) 465–476.
[6] R.G. Downey, M.R. Fellows, Parametrized Complexity, Springer, Berlin, 1999.
[7] W. Farmer, Simple second order languages for which unification is undecidable, J. Theor. Comput. Sci. 87 (1991) 173–214.
[8] W. Farmer, A unification algorithm for second order monadic terms, Ann. Pure Appl. Logic 39 (1988) 131–174.
[9] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W.H. Freeman and Co., San Francisco, 1979.
[10] W.D. Goldfarb, The undecidability of the second order unification problem, Theor. Comput. Sci. 13 (1981) 225–230.
[11] C. Gutierrez, Satisfiability of word equations with constants is in exponential space, in: Proceedings FOCS'98, IEEE Computer Society Press, Palo Alto, California, 1998, pp. 112–119.
[12] G. Huet, A unification algorithm for typed λ-calculus, Theor. Comput. Sci. 1 (1975) 27–57.
[13] J. Levy, Decidable and undecidable second order unification problems, in: Proceedings of the 9th International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 1379, 1998, pp. 47–60.
[14] J. Levy, M. Veanes, On the undecidability of second order unification, Informat. Comput. 159 (2000) 125–150.
[15] J. Levy, Linear second order unification, in: Proceedings of the 7th International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 1103, 1996, pp. 332–346.
[16] G. Makanin, The problem of solvability of equations in a free semigroup, Math. USSR Sbornik 32 (2) (1977) 129–198.
[17] W. Plandowski, Satisfiability of word equations with constants is in NEXPTIME, in: T. Leighton (Ed.), Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC'99), ACM Press, Atlanta, Georgia, 1999, pp. 721–725.
[18] W. Plandowski, Satisfiability of word equations with constants is in PSPACE, in: FOCS 99, 1999, pp. 495–500.
[19] M. Schmidt-Schauß, K.U. Schulz, On the exponent of periodicity of minimal solutions of context equations, in: Proceedings of the 9th International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 1379, 1998, pp. 61–75.
[20] M. Schmidt-Schauß, K.U. Schulz, Solvability of context equations with two context variables is decidable, in: Proceedings of the International Conference on Automated Deduction, Lecture Notes in Computer Science, vol. 1632, 1999, pp. 67–81.
[21] M. Schmidt-Schauß, K.U. Schulz, Solvability of context equations with two context variables is decidable, J. Symbolic Comput. 33 (1) (2002) 77–122.
[22] M. Schmidt-Schauß, Unification of stratified second order terms, Internal Report 12/94, Fachbereich Informatik, J.W. Goethe-Universität Frankfurt, Frankfurt, Germany, 1994.
[23] M. Schmidt-Schauß, A decision algorithm for distributive unification, Theor. Comput. Sci. 208 (1998) 111–148.
[24] M. Schmidt-Schauß, A decision algorithm for stratified context unification, J. Logic Comput. 12 (6) (2002) 929–953.
[25] W. Snyder, J. Gallier, Higher order unification revisited: complete sets of transformations, J. Symbolic Comput. 8 (1989) 101–140.
[26] D.A. Wolfram, The Clausal Theory of Types, no. 21 in Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, 1993.